DOCTEUR EN SCIENCES

Recent developments in optimization methods for data assimilation in oceanography

Laloyaux, Patrick

Award date:
2012

Awarding institution:
Universite de Namur

Facultés universitaires Notre-Dame de la Paix
Faculté des Sciences — Département de Mathématique

Recent developments in optimization methods for data assimilation in oceanography

Dissertation présentée par
Patrick Laloyaux
en vue de l'obtention du grade
de *Docteur en Sciences*

Composition du jury :

Timoteo Carletti
Serge Gratton
Anne Lemaitre
Annick Sartenaer
Luís Nunes Vicente
Arthur Vidard

2012

Imprimé en Belgique
ISBN : 978-2-87037-770-3
Dépôt légal : D / 2012 / 1881 / 33

Facultés universitaires Notre-Dame de la Paix
Faculté des Sciences
Rue de Bruxelles 61, B-5000 Namur, Belgium

Facultés universitaires Notre-Dame de la Paix
Faculté des Sciences
Rue de Bruxelles 61, B-5000 Namur, Belgium

Développements de nouvelles méthodes d'optimisation pour l'assimilation de données en océanographie

Patrick Laloyaux

Résumé. — Ce travail concerne le développement et l'étude de nouveaux algorithmes d'optimisation pour la résolution de problèmes d'assimilation de données en océanographie. L'équivalence théorique entre le filtre SEEK et une variante réduite du problème 4D-Var permet de proposer deux méthodes innovantes de minimisation pour l'approche variationnelle. La première développe un point de départ adéquat et un préconditionneur à mémoire limitée pour accélérer une méthode de type gradient conjugué, utilisée dans le procédé incrémental. La seconde propose une approche sans dérivée qui minimise la fonction 4D-Var en construisant et en explorant une séquence de sous-espaces appropriés de dimension faible. Les deux méthodes ont été analysées numériquement sur des modèles académiques en eaux peu profondes avant d'être évaluées sur la configuration GYRE du logiciel NEMO.

Recent developments in optimization methods for data assimilation in oceanography

Patrick Laloyaux

Abstract. — This work is concerned with the development and the study of novel optimization algorithms for solving data assimilation problems with application in oceanography. Motivated by the theoretical equivalence between the SEEK filter and a reduced variant of the 4D-Var problem, two innovative minimization methods for the variational approach are proposed. The first one derives an appropriate starting point and a limited-memory preconditioner to accelerate the conjugate gradient-like algorithm used in the incremental method. The second one proposes a derivative-free approach which minimizes the 4D-Var function by constructing and exploring a sequence of appropriate low dimensional subspaces. Both methods have been numerically analyzed on academical shallow water models before being assessed on the GYRE configuration of the NEMO framework.

Dissertation doctorale en Sciences (orientation mathématique)
(Ph.D. thesis in Mathematics)
Date : 28/09/2012
Département de Mathématique
Promoteur (advisor) : Pr. A. Sartenaer

Remerciements

Cette thèse n'aurait pu voir le jour sans le concours de nombreuses personnes qui m'ont guidé, soutenu et encouragé tout au long de mon doctorat.

En premier lieu, je tiens à remercier ma promotrice, Annick Sartenaer, de m'avoir accueilli au sein de l'Unité d'Analyse Numérique et aidé tout au long de cette thèse. Je la remercie pour le soutien qu'elle m'a apporté dans le développement de mes collaborations et pour ses conseils lors de travaux de rédaction. Je la remercie également de m'avoir donné l'opportunité de participer à plusieurs conférences internationales. Ce fut pour moi une chance d'y présenter mes travaux et d'y rencontrer de nombreux chercheurs. Je souhaite exprimer ma reconnaissance à Serge Gratton pour les nombreux conseils qu'il m'a prodigués ainsi que pour le temps qu'il m'a consacré tout au long de ce projet. Nos discussions ont toujours été très enrichissantes et m'ont permis d'avancer dans mes travaux de recherche. Je tiens à remercier aussi Arthur Vidard pour la sympathie de son accueil et pour m'avoir permis de travailler sur un modèle océanographique opérationnel. Mes remerciements vont aussi aux autres membres du jury : Timoteo Carletti, Anne Lemaitre et Louís Nunes Vicente pour leurs conseils pertinents et leurs remarques constructives lors de la défense privée de ma thèse.

Je remercie Jean Tshimanga pour sa présence dans mon comité d'accompagnement et pour son suivi en début de thèse. Son expérience et sa disponibilité ont été un grand atout lors de mon lancement dans le domaine de l'assimilation de données. Je tiens à remercier sincèrement Philippe Toint pour m'avoir permis d'acquérir une bonne formation en optimisation à travers ses cours et lors de mon mémoire de fin de Master. Pendant ma thèse, j'ai séjourné principalement dans deux centres de recherche étrangers. Tout d'abord, au CERFACS, à Toulouse, dans l'équipe Parallel Algorithms. Je tiens à remercier Ian Duff, Serge Gratton et Xavier Vasseur de m'avoir accueilli. Je garde un excellent souvenir de la sympathie des membres de cette équipe. Ensuite, au laboratoire Jean Kutzmann, à Grenoble, dans l'équipe MOISE. Je remercie Eric Blayo et Arthur Vidard de m'avoir accueilli. Je remercie Claire Chauvin d'avoir toujours trouvé du temps pour répondre à mes multiples questions.

Je remercie chaleureusement celles et ceux qui m'ont accompagné au quo-

tidien : mes collègues de bureau. Merci à Johan Barthélemy, Laurie Hollaert, Vincent Malmedy et Laetitia Legrain. Je voudrais remercier particulièrement Vincent pour avoir répondu à bon nombre de mes questions en informatique et en mathématiques. Merci aussi aux autres scientifiques de l'équipe d'analyse numérique : Charlotte Tanier, Dimitri Tomanos et Melissa Weber Mendonça, notamment pour les bons moments partagés lors des conférences. D'une manière plus générale, je remercie tous les membres du département pour la bonne ambiance qui y règne pendant les pauses et dans les couloirs.

Pendant ces cinq années, j'ai pu partager l'encadrement de certains travaux pratiques avec d'autres assistants. Je tiens à remercier Nicolas Delsate et Julien Dufey pour la bonne collaboration ainsi que les bons moments passés après les cours. Je tiens à remercier Wim Vanhoof pour avoir réussi à me transmettre sa passion pour la programmation lorsque j'étais étudiant et pour sa sympathie et la bonne collaboration lorsque je suis devenu son assistant pour l'un de ses cours. Je tiens aussi à remercier Jean-Pierre Mbungu, Mukenga Bantu Kazadi et Kavul Mutshail pour l'accueil qu'ils m'ont réservé lors de mon séjour à l'UNIKIN, à Kinshasa, comme professeur invité. Merci à Johan Barthélemy, Jehan Boreux, Audrey Compère et Manuel Peremans pour les bons moments passés pendant et après nos études. Je leur souhaite mes meilleurs voeux de réussite pour la suite.

Je tiens à remercier toute ma famille (s'agrandissant d'année en année) pour le soutien qu'elle m'apporte jour après jour et ce depuis toujours.

Enfin mon dernier remerciement s'adresse à Sara. Merci d'avoir été à mes côtés pendant la longue et difficile période de la rédaction de cette thèse. Merci pour tout ce que tu m'apportes quotidiennement. Un beau défi anglais nous attend.

A tous, encore merci.

Patrick

Contents

Introduction

To compute marine and weather forecasts, dynamical systems are integrated on high performance computers within a given time window starting from an initial state (also called initial condition). This time window is related to the relevant time scales of the considered physical situation. The ability to make an accurate forecast requires both that the dynamical system be a good representation of the reality and that the initial condition be known precisely. Numerical weather forecasts are performed since the 1950s and have witnessed an explosion of activity since the 1980s. Over the years, the quality of the models and methods for using observation has improved continuously, resulting in major forecast advancements. These progresses are related to the increased amount of observational data and to the increased power of supercomputers that is available. The problem of determining the initial condition for a forecast is very complex and has become an active area of research on its own. *Data assimilation methods aim at specifying an initial condition of a dynamical system by combining both the information from a numerical model and from observations* (see Kalnay, 2003; Rabier, 2005, for a review). There are two main approaches to solve data assimilation problems. On the one hand, the sequential approach is based on statistical estimation theory and contains the Kalman filter algorithm with its numerous adaptations. On the other hand, the variational approach, based on optimal control theory, proposes a family of optimization problem formulations with minimization techniques. Among those ones, the 4D-Var formulation with its incremental technique is the most famous one and is used nowadays operationally in weather forecasting centers.

This thesis is concerned with the design, the implementation and the assessment of novel optimization algorithms for solving data assimilation problems with application in oceanography. In the sequential approach, the Singular Evolutive Extended Kalman (SEEK) filter is a modification of the Kalman filter which reduces the dimension of the estimator error space thanks to the use of Empirical Orthogonal Functions (EOFs). This reduction technique allows to solve large data assimilation problems and has been successfully implemented in operational oceanographic data assimilation frameworks. In this thesis, we demonstrate a connection between the SEEK filter and a reduced variant of the

4D-Var problem. Motivated by this theoretical result, the reduction technique based on the EOFs is transferred from the sequential to the variational approach and two innovative minimization methods for the variational approach are developed. The first minimization method derives an appropriate starting point and a limited-memory preconditioner which may accelerate the minimization process of the 4D-Var function. The second one proposes a derivative-free approach which minimizes the 4D-Var function by constructing and exploring a sequence of appropriate low dimensional subspaces. Both methods have been numerically analyzed on academical shallow water models first before being assessed on the operational ocean model NEMO.

Structure of the thesis

We next give a brief description of the content of the thesis. The main contributions are highlighted at the end of the manuscript.

Chapter 1 gives an historical introduction to weather forecasting. Moreover, it illustrates the importance of such forecasts by listing their most common applications and exposes some future challenges. The next chapter presents the basic concepts of data assimilation problems. It states the terminology used throughout this work and the mathematical materials useful for our needs. Chapter 3 gives a survey of the main variational formulations with a broad outline about the techniques to minimize them. Chapter 4 presents the sequential approach with its well-known Kalman filter. A flops counting shows that this filter is not affordable for large data assimilation problems and the SEEK filter is then derived. These surveys of variational and sequential approaches are far from being exhaustive but essentially cover tools that will be used and discussed in the following chapters. The well-known connection between the Kalman filter and the 4D-Var problem in the context of linear model and observation operators are demonstrated in Chapter 5. This result is used to establish a connection between the SEEK filter and a reduced variant of the 4D-Var problem based on the use of the EOFs. This connection is the key stone to motivate the development of our two new minimization methods for the variational approach. On the one hand, Chapter 6 describes an enhanced incremental method where an appropriate starting point and a limited-memory preconditioner have been derived from information contained in the EOFs. A shallow water model is used to illustrate the positive impact produced on the convergence rate of the incremental method. On the other hand, Chapter 7 gives an attempt to develop a derivative-free approach for solving the 4D-Var problem. It minimizes the 4D-Var function by building and exploring a sequence of appropriate low dimensional subspaces constructed from EOFs information. This approach is numerically compared with the Ensemble Kalman filter (EnKF) which is a Monte-Carlo, derivative-free, implementation of the Kalman filter. Our derivative-free approach shows its efficiency on a shallow

water model by producing estimates of comparable quality with those produced by EnKF. Finally, Chapter 8 illustrates the numerical behavior of our two minimization methods on the operational ocean model NEMO.

We finally conclude and give some perspectives of future work. A summary of our contributions and tables containing the main notations used are also presented after the conclusion.

Chapter 1

History and motivations

1.1 History

Human beings have attempted to predict the weather informally for millenniums. The Babylonians were among the first to attempt weather forecast in 650 BC. Their predictions were based on the appearance of cloud patterns and folk wisdom. Later, weather captured the imagination of the greatest minds of the Renaissance. Galileo Galilei invented a thermometer in the 16th century, and Evangelista Torricelli a mercury barometer in the 17th century. Such instruments enabled quantitative observations of the atmosphere. The invention of the electric telegraph in the beginning of the 19th century allowed to broadcast weather conditions from a wide area almost instantaneously. These technological progresses made a giant leap toward modern meteorology.

In 1890, the American meteorologist Cleveland Abbe recognized that meteorology is essentially the application of hydrodynamics and thermodynamics to the atmosphere. For his part, the Norwegian physicist stated in 1904 two necessary conditions for rational forecasting. Firstly, the governing laws of the atmosphere must be correctly modeled. Secondly, the state of the atmosphere must be known accurately at a given time. These two conditions are still the base of our actual weather forecast. Unfortunately, their ideas were not applicable in practice since no reliable numerical method existed to solve systems of nonlinear partial differential equations and since analytical solutions were infeasible. Few years later, the English mathematician Lewis Fry Richardson developed numerical methods for solving partial differential equations using a spatial and temporal discretization. He manually computed one weather forecast which took him six weeks of work. The results were disastrous but his brilliant idea is the foundation of modern forecasting. He published in 1922 a book with some visionary ideas. The most famous describes a forecast factory that carries out the process of calculating weather. In fact, without realizing

it, he described a massively parallel processing even before the invention of a digital computer:

«Imagine a large hall like a theater, except that the circles and galleries go right round through the space usually occupied by the stage. The walls of this chamber are painted to form a map of the globe. The ceiling represents the north polar regions, England is in the gallery, the tropics in the upper circle, Australia on the dress circle and the Antarctic in the pit.

A myriad computers are at work upon the weather of the part of the map where each sits, but each computer attends only to one equation or part of an equation. The work of each region is coordinated by an official of higher rank. Numerous little "night signs" display the instantaneous values so that neighboring computers can read them. Each number is thus displayed in three adjacent zones so as to maintain communication to the North and South on the map.

From the floor of the pit a tall pillar rises to half the height of the hall. It carries a large pulpit on its top. In this sits the man in charge of the whole theater; he is surrounded by several assistants and messengers. One of his duties is to maintain a uniform speed of progress in all parts of the globe. In this respect he is like the conductor of an orchestra in which the instruments are slide-rules and calculating machines. But instead of waving a baton he turns a beam of rosy light upon any region that is running ahead of the rest, and a beam of blue light upon those who are behindhand.

Four senior clerks in the central pulpit are collecting the future weather as fast as it is being computed, and dispatching it by pneumatic carrier to a quiet room. There it will be coded and telephoned to the radio transmitting station. Messengers carry piles of used computing forms down to a storehouse in the cellar.

In a neighboring building there is a research department, where they invent improvements. But these is much experimenting on a small scale before any change is made in the complex routine of the computing theater. In another building are all the usual financial, correspondence and administrative offices. Outside are playing fields, houses, mountains and lakes, for it was thought that those who compute the weather should breathe of it freely» (Richardson, 1922). Richardson's idea is represented in Figure 1.1 by the Belgian artist François Schuiten.

In the early 1950's, the development of digital computers provided a mean of applying Richardson's methods and attacked the enormous computational task involved in weather forecasting. The first numerical weather prediction has been computed on the Electronic Numerical Integrator and Computer (ENIAC) constructed by John von Neumann, a Hungarian-American mathematician. One can remark that the four most important scientists who had marked the beginning of weather forecasting come from four different countries (USA, Norway, UK and Hungary) with an expertness in four different fields (Meteorology, Mathematics, Physics and Computation). The field of weather

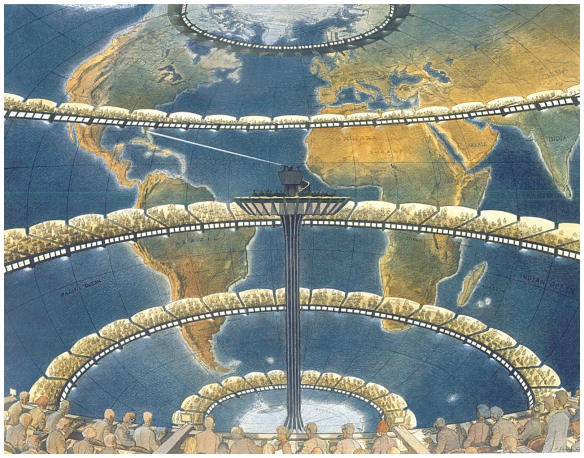

Figure 1.1 — *An artist's impression of Richardson's forecast factory (©François Schuiten).*

forecasting is a beautiful illustration of an international and multidisciplinary scientific cooperation (see Wiin-Nielsen, 1991, Lynch, 2008, for more historical details).

During the six past decades, the development of weather forecasting can be mainly sorted in four different ways. Firstly, tremendous improvements in the modelization of atmosphere and ocean have been done. General circulation models with improved representation of physical processes, using sophisticated modeling techniques, have been developed. In parallel, numerical methods used to compute the evolution of such models have been constantly improved with progresses in implicit integration schemes allowing longer time steps. Secondly, technological breakthroughs with the invention of the radiosonde and later of the satellites have been crucial for collecting large set of accurate observations and better understanding of the earth system. Thirdly, data assimilation methods which produce an estimation of the system using both information from a model and observations have grown up and became a science in itself. Whereas basic interpolation methods were used for the first weather forecasts, more sophisticated methods based on optimal control and statistical theories are deployed nowadays. Finally, the power of computers has been constantly increased with the invention of vector processors and massively parallel computers. It allows to increase the size of the domains as well as the spatial and temporal discretizations which enable to represent small-scale phenomena. A significant part of the 500 most powerful computers (`http://www.top500.org`) are devoted to wheater forecasting showing that this activity use supercomputers at the cutting edges of the progress.

Nowadays, most of the industrialized countries develop national weather

forecasting centers for both operational predictions and scientific researches. They run a large set of models which simulate not only the ocean and the atmosphere but also geophysical, chemical and biological processes. These models are usually coupled together to transfer information among them and produce a wide range of marine and atmospheric forecast for different time and space scales.

1.2 Motivations

Several countries employ government agencies to provide weather and marine forecast. The first aim of such agencies is to give warnings and advisories to the public in order to protect their life and property. The second one is to maintain commercial interests of companies (Katz and Murphy, 1997). In this section, few examples for both cases are given to show the crucial importance of forecasting in our daily life.

A major part of modern weather forecasting is to issue alerts in the case of severe or hazardous weather. Some of the most commonly alerts are for thunderstorms, high winds, floods, snow and ice conditions. New alerts are constantly developed to warn the public when, for example, the intensity of the ultraviolet radiation is high, the risk of avalanche is important, the coastal waves are high or a risk of tsunami after an submarine earthquake is present.

The second aim of weather forecasting which deals with commercial interests is even more diversified. Because the aviation industry is especially sensitive to the weather, accurate atmosphere forecasting is essential. Indeed, some weather conditions such as fog, turbulence, icing or thunderstorms can increase in-flight hazards. Volcanic ash is also a significant problem for aviation, as aircraft can lose engine power within ash clouds. Additionally, airports often decide which runway is being used to take advantage of a headwind since it reduces the distance required for takeoff, and eliminates potential crosswinds. At last, an accurate knowledge of jet streams can save fuel using them. The power companies need also accurate temperature forecast since they use them to determine how many units of energy to produce and/or to buy. Too few or too many units of produced energy will result in a waste of money and ultimately in a higher cost to the consumer. On their sides, the agriculture and food industries are interested by rain amount, temperature (especially freezing and very hot temperatures) and long term drought or heavy rain to optimize their production effectiveness. Other companies are more interested by marine forecasting. Commercial ships are interested by ocean currents, wind direction and speed and wave heights since they influence the safety and the speed of marine transit. On their sides, deep sea oil companies pay attention of wave and ocean currents to ensure stability of their off-shore platforms.

1.3 Challenges

Meteorologists have remarkably increased the accuracy of weather forecast by assimilating more observations with suitable models and data assimilation methods. On average, a five-day weather forecast of today is as reliable as a two-day weather forecast 20 years ago. Despite this major scientific progress, many challenges remain especially for medium and long term weather predictability. Indeed, weather forecast is a complex field because the atmosphere and the ocean are complicated systems that are affected by many factors and can react in different ways. These systems are chaotic, meaning that small differences in initial conditions yield widely diverging outcomes. In this context, long-term predictions are complex. A first challenge is to continue the development of new methods which increase the accuracy in the determination of initial conditions and to better estimate the errors in the produced forecast. These methods should appropriately predict severe weather and extreme climate events such as heavy rainfall and cyclones. Besides that, there is a growing societal demand for environmental predictions. This demand can be matched by using a global approach which models a whole set of physical, chemical and biological processes for the earth system. In such a situation, new data assimilation methods are needed. They may be able to use coupled models with different time and space scales and to transfer information among the different model components.

Chapter 2

Data assimilation problems

Mathematical models are critical to understand and predict the behavior of real-world complex systems such as the atmosphere and the ocean. Forecasts of these systems are computed by integrating these models on high performance computers, starting from an *initial state* (also called *initial condition*). The accuracy of the forecast depends on the ability of the mathematical models to adequately represent the reality and on the knowledge of an accurate initial state. The determination of an accurate initial condition using observational data of the system is called a *data assimilation problem* (Bennett, 2002, Kalnay, 2003). Even if data assimilation problems arise from very different fields, they share the same materials. These materials are presented in this chapter. For consistency with the literature on the subject, the notation proposed by Ide et al. (1997) will be adopted as far as possible.

2.1 The state vector

The first step in the mathematical formalization of a data assimilation problem is the definition of the work space. Each system is characterized by the values of its physical fields which are often continuous functions. Nevertheless, for reasons that will be presented in the next section, a discretization is used to represent the state of a system. A vector $\mathbf{x} \in \mathbb{R}^n$ is defined which contains the values of the physical fields at the discretization points. Such a vector is called a *state vector*, where n represents its size. As an example, if the system under study is the ocean or a part of it, the state vector could contain the value of some physical fields such as the temperature, the salinity, the velocities, etc., at different places of the ocean.

2.2 The model

Mathematical models are derived by applying laws of physics. As an example, the motion of the ocean can be modeled using the Navier-Stokes equations, while the evolution of the atmosphere can be modeled using the Newton's second law, the continuity equation, the equation of state for ideal gases, the first law of thermodynamics and a conservation equation for water mass (see, Bjerknes, 1904)). In general, one can assume that a mathematical model can be represented by a set of Ordinary Differential Equations (ODEs) or Partial Differential Equations (PDEs). Unfortunately, the solution of such systems of equations can generally not be solved analytically and only numerical solutions are available.

Different methods can be applied to compute numerical solutions. A first possibility is to perform a discretization in time and a discretization in space using a mesh on the spatial domain. The most simple mesh is a uniform grid but other, more complex, meshes such as curvilinear grids or adaptive mesh grids are possible. Once a mesh is defined, a numerical method such as the leapfrog scheme can be used to find a numerical solution of the PDEs system (Cushman-Roisin and Beckers, 2010). Another possibility to solve PDEs systems is to use a finite element method which approximates the exact solution using polygons or tetrahedra as spatial discretization and a basis of piecewise linear functions. The numerical solution is a linear combination of the basis functions where the coefficients are computed by solving a set of ODEs using standard techniques such as the Runge-Kutta method. In this case the state vector does not contain the values of the physical fields at grid points anymore but the coefficients of the linear combination of the basis functions. The size of the state vector is the number of basis functions used for the approximation. Finally, a last usual method to solve PDEs systems is the spectral method. It is similar to that of the finite element method but uses spherical harmonics as basis functions. In any case, the spatial discretization or the finite number of basis functions implies to use a state vector of finite dimension.

Mathematically, the evolution of the model from time t_i to time t_{i+1} is represented by a *model operator*

$$\mathcal{M}_{i+1,i} : \mathbb{R}^n \to \mathbb{R}^n : \mathbf{x}_i \rightsquigarrow \mathbf{x}_{i+1} = \mathcal{M}_{i+1,i}\mathbf{x}_i, \tag{2.1}$$

where $\mathbf{x}_i \in \mathbb{R}^n$ is a state vector representing the state of the system at time t_i and $\mathbf{x}_{i+1} \in \mathbb{R}^n$ is a state vector representing the state of the system at time t_{i+1}. Calligraphic font is used whenever the considered operator is assumed to be nonlinear. In our entire work, the model is assumed to be perfect, which means that no randomness is added to the model evolution. If the model operator is linear, the dynamical system is described by

$$\mathbf{x}_{i+1} = \mathbf{M}_{i+1,i}\mathbf{x}_i, \tag{2.2}$$

where $\mathbf{M}_{i+1,i} \in \mathbb{R}^{n\times n}$ is an n by n matrix.

2.3 The Observations

When studying an earth's system, observations of different physical fields are often needed. For example, considering the atmosphere, observations of air speed, density, pressure, temperature and humidity are crucial. These kind of observations can be collected by balloons, planes or weather stations and are called *in situ observations* since they are directly measured. Considering now the ocean, buoys (moored or drifting) and ships can provide in situ observations for physical fields such as sea surface temperatures, salinity, sea level or ocean currents. The quality and the distribution of in situ measurements are non-uniform across space and time. As an example of observation system, Argo is a collection of small, drifting oceanic robotic probes deployed worldwide. The probes float as deep as 2 km measuring salinity and temperature profiles. Once every 10 days, the probes surface and transmit measurements to scientists on shore. This system deploys $3,500$ probes which is a large number for an operational observation system. Nevertheless, since they are distributed for the whole ocean, the measurements are sparse in space and time (see Figure 2.1).

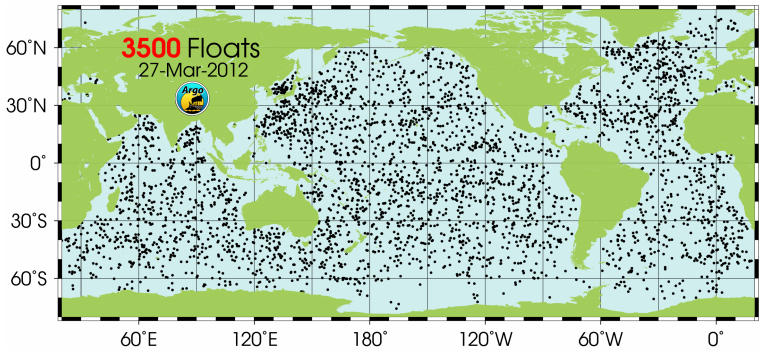

Figure 2.1 — *Distribution of Argo buoys (©Argo Information Centre).*

In situ measurements provide detailed conditions in specific locations, but to obtained more measurements distributed in space and time, scientists are relying on satellites. In few days, satellites provide uniform observations on the entire surface of the globe. These observations are renewed permanently. The speed of a satellite is about 1500 times that of a ship, so it is able to observe the entirety of the ocean rather quickly. They allow in particular the observation of high latitudes for which few in situ measurements exist. The satellites are able to measure, amongst others, the temperature of the ocean surface, the wind speed, the wind direction, the wave height, the wave direction, the sea level or the extent of sea ice. For example, the QuikSCAT satellite measures near-surface wind speed and wind direction over oceans under all weather and cloud

conditions. The wind speed measurements are performed with an accuracy of 2 meters/second, and the wind direction, with an accuracy of 20 degrees. The QuikSCAT satellite observes oceans on a 1,800-kilometer-wide band with a resolution of 25 kilometers, making approximately 400,000 measurements and covering 90% of the Earth's surface in one day (see Figure 2.2). It is however

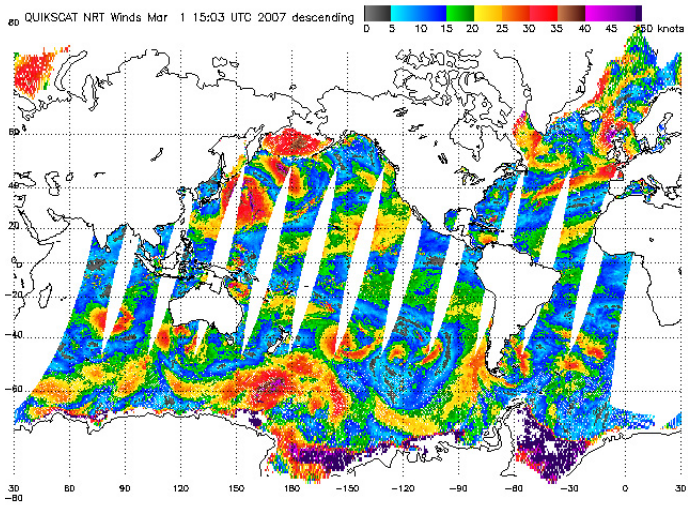

Figure 2.2 — *A sequence of descending passes of the QuikSCAT satellite measuring wind speed and wind direction (©National Oceanic and Atmospheric Administration).*

not yet possible to suppress in situ measurements since satellites cannot observe any physical field and cannot perform observations in deep waters.

Mathematically, an observation of the system performed at time t_i and containing p_i measurements is represented by a vector $\mathbf{y}_i \in \mathbb{R}^{p_i}$ defined as

$$\mathbf{y}_i = \mathcal{H}_i \mathbf{x}_i^t + \boldsymbol{\epsilon}_i^o, \tag{2.3}$$

where $\mathbf{x}_i^t$ is the true (unknown) state vector at time t_i, $\boldsymbol{\epsilon}_i^o \in \mathbb{R}^{p_i}$ is the observational noise due to instrumental and representativeness errors and $\mathcal{H}_i$ is the *observation operator* which maps any state vector into the observation space $\mathbb{R}^{p_i}$ as

$$\mathcal{H}_i : \mathbb{R}^n \to \mathbb{R}^{p_i} : \mathbf{x}_i \rightsquigarrow \mathcal{H}_i \mathbf{x}_i. \tag{2.4}$$

As for the model operator, calligraphic fonts are used whenever the considered operator is assumed to be nonlinear. When the observation operators are linear, an observation is obtained by

$$\mathbf{y}_i = \mathbf{H}_i \mathbf{x}_i^t + \boldsymbol{\epsilon}_i^o, \tag{2.5}$$

where $\mathbf{H}_i \in \mathbb{R}^{p_i \times n}$ is a p_i by n matrix which is assumed to be of full rank in our entire work. An observation operator is needed for two main reasons. Firstly, measurements are not necessarily performed for each component of the state vector and they can be made at different locations than the grid points. Secondly, the physical fields can be observed indirectly. In the following, we briefly illustrate both reasons and show at the same time how nonlinearities appear in the observation operator.

2.3.1 Interpolation

The problem of measurements obtained at locations which are not grid points can be solved by using interpolation techniques to map the state vector in the observation space. We give an example of such a technique in the following. Assume that a measurement is available at point Q on a two-dimensional grid (see Figure 2.3) with coordinates (a, b) in a Cartesian plane. One simple possibility is to perform a bilinear interpolation using the four grid points closest to the observation location, $P_{11} = (a_1, b_1)$, $P_{12} = (a_1, b_2)$, $P_{21} = (a_2, b_1)$ and $P_{22} = (a_2, b_2)$, with their state values, say, $f(a_1, b_1)$, $f(a_1, b_2)$, $f(a_2, b_1)$ and $f(a_2, b_2)$, respectively. The bilinear interpolation first performs two linear in-

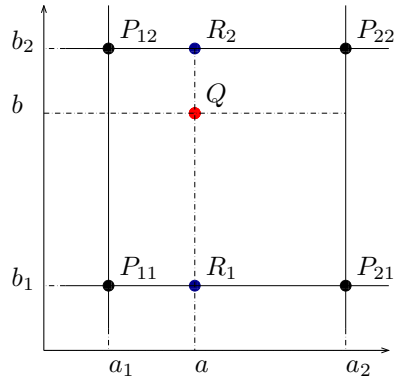

Figure 2.3 — *Bilinear interpolation scheme.*

terpolations along the horizontal axis to approximate the value of the state vector at points $R_1 = (a, b_1)$ and $R_2 = (a, b_2)$. These approximations are given by

$$f(a, b_1) \approx \frac{a_2 - a}{a_2 - a_1} f(a_1, b_1) + \frac{a - a_1}{a_2 - a_1} f(a_2, b_1) \tag{2.6}$$

and

$$f(a, b_2) \approx \frac{a_2 - a}{a_2 - a_1} f(a_1, b_2) + \frac{a - a_1}{a_2 - a_1} f(a_2, b_2). \tag{2.7}$$

Using the interpolated values $f(a, b_1)$ and $f(a, b_2)$ at the points R_1 and R_2, a linear interpolation can is then performed along the vertical axis to approximate

the value of the state vector at $Q = (a, b)$

$$f(a,b) \approx \frac{b_2 - b}{b_2 - b_1} f(a, b_1) + \frac{b - b_1}{b_2 - b_1} f(a, b_2). \tag{2.8}$$

Using (2.6) and (2.7) in (2.8), a general formulation of this bilinear interpolation process can be derived, yielding

$$f(a,b) \approx \frac{f(a_1,b_1)}{(a_2-a_1)(b_2-b_1)}(a_2-a)(b_2-b) + \frac{f(a_2,b_1)}{(a_2-a_1)(b_2-b_1)}(a-a_1)(b_2-b)$$
$$+ \frac{f(a_1,b_2)}{(a_2-a_1)(b_2-b_1)}(a_2-a)(b-b_1) + \frac{f(a_2,b_2)}{(a_2-a_1)(b_2-b_1)}(a-a_1)(b-b_1).$$

Note that the bilinear interpolation is not a linear but a quadratic function, except for observations along the mesh grid. Figure 2.4 shows the contour lines of the interpolated values for a unit square box with $f(0,0) = 0$, $f(0,1) = 1$, $f(1,0) = 1$ and $f(1,1) = 0.5$, showing the nonlinear behavior of the bilinear interpolation scheme. For a three dimensional field, a trilinear interpolation

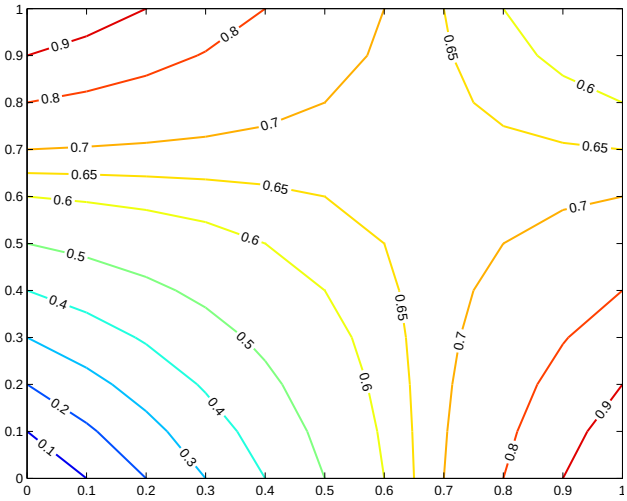

Figure 2.4 — *Example of a bilinear interpolation showing the nonlinear behavior.*

exists which is a simple extension of the bilinear interpolation to three dimensions.

2.3.2 Indirect observations

As said before, an observation operator can also be needed if the physical field is not directly observed. As an example, we assume that the state vector $\mathbf{x}_i$

contains only the value of the sea surface temperature (SST) at one location and that a satellite is used to perform a measurement at this location such that no interpolation is needed. In practice, the satellites are not able to directly measure SST but they measure radiation using infrared radiometers (Pimentel, 2006). In this case, $\mathbf{y}_i$ contains one radiation measurement while $\mathbf{x}_i$ contains a SST value. A model operator is needed to transform SST values into radiation values. A possibility is to use the relation

$$\mathbf{y}_i = \alpha\sigma\mathbf{x}_i^4(0.39 - 0.05\sqrt{\kappa})(1 - \delta_c c^2) + 4\alpha\sigma\mathbf{x}_i^3(SST - T_a) \tag{2.9}$$

where α is the surface emissivity, σ is the Stefan-Boltzmann constant, κ is the water vapor pressure in surface air, c is the fractional cloud cover, δ_c is the lattice dependent cloud cover coefficient and T_a is the air temperature (see Clark et al., 1974, for details). This kind of relation is modeled in the observation operator $\mathcal{H}_i$. We can see that the relation is nonlinear. It shows another origin of nonlinearities in the observation operators. Of course, other possibilities exist to deduce SST from satellite observations but they will not be presented in this work since the goal was simply to illustrate the fact that the observation operator can be use to transform physical fields.

2.3.3 Observation errors

A very large variety of sensors are employed for operational data collection schemes. These sensors operate in a wide range of environments and use different measurement protocols. The observation error vector $\boldsymbol{\epsilon}_i^o \in \mathbb{R}^{p_i}$ is a random vector which models the error that could be made by the observation process. The origin of error is twofold and can be illustrated using the previous example of a radiation measurement performed by a satellite. Firstly, an instrumental errors can occur, due to a measurement error in the radiation. Secondly, a representativeness error can occur, due to the approximation made in a relation such as (2.9).

It is not trivial to estimate the observation error $\boldsymbol{\epsilon}_i^o$. In our work, the mean of the observation error is assumed to be zero and the *observation error covariance matrix*, $\mathbf{R}_i \in \mathbb{R}^{p_i \times p_i}$, is assumed to be sparse symmetric positive definite. Moreover, the observation errors are assumed to be temporally uncorrelated. The covariance matrix $\mathbf{R}_i$ can be decomposed as

$$\mathbf{R}_i = \boldsymbol{\Sigma}\mathbf{C}\boldsymbol{\Sigma},$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{p_i \times p_i}$ is a diagonal matrix which contains the standard deviation of each measurement and $\mathbf{C} \in \mathbb{R}^{p_i \times p_i}$ is a correlation matrix. The standard deviations are often estimated using the characteristics of the measurement devices. They must also include the representativeness error due to the fact that some physical phenomena are not represented in the observation operator. The correlation matrix is often assumed to be the identity, meaning that no spatial correlation between measurements are modeled.

2.3.4 Observation selection

Data assimilation problems are mathematical problems that are usually solved iteratively. A set of observations is available from the past and is continuously updated with new measurements from the present. Nevertheless, the whole set of observations is not used to produce an estimation of the system's state but only new observations from a relevant area and time interval are used to improve the previous estimation made from past observations. The time interval used to collect the new observations is called the *data assimilation window*. For atmospheric systems, its typical range is between 6 to 24 hours, while between several days to one month for oceanographic systems. Formally, the time interval defined by the assimilation window is given by $[t_0, t_N]$ and the number of measurements used during this period is given by

$$p = \sum_{i=1}^{N} p_i,$$

where N is the number of observation times. A new observation vector $\mathbf{y} \in \mathbb{R}^p$ which concatenates the N different ones can be defined as

$$\mathbf{y} = \left((\mathbf{y}_1)^{\mathrm{T}}, \ldots, (\mathbf{y}_N)^{\mathrm{T}} \right)^{\mathrm{T}}.$$

Its size p is usually much smaller than the size of the state vector n. Nowadays, weather forecasting centers use several millions of observations each day to estimate the state of their atmospheric and oceanographic models which can contained up to a billion of unknown parameters. The selection of the observations which are distributed spatially all around the world is not straightforward. In Figure 2.5, the areas where observations are collected for a 24-h, a 72-h and a 120-h atmospheric forecast for the central zone Z is shown. One can observe

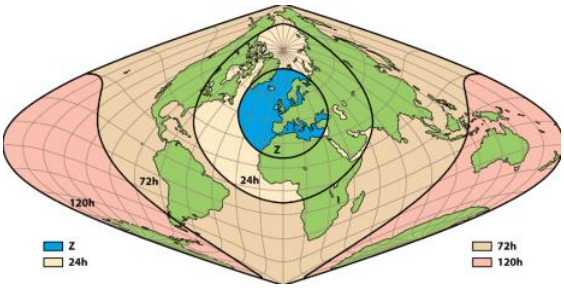

Figure 2.5 — *Areas where observations are collected for a 24-h, a 72-h and a 120-h forecast in the central zone Z (©European Centre for Medium-Range Weather Forecasts).*

that for a five days forecast in the central zone, observations from all around the world must be collected.

2.4 The Background

As said before, the estimation of the state of a system is usually computed iteratively. The previous estimation made from past observations is called the *background vector* and is denoted $\mathbf{x}^b \in \mathbb{R}^n$. This vector is available at the beginning of each data assimilation window, thus at time t_0. The background error is defined as the error between the background vector and the related true (unknown) state vector. In our work, we assume that the background error is unbiased and uncorrelated with the observations error. We suppose also that the *background error covariance matrix*, $\mathbf{B} \in \mathbb{R}^{n\times n}$, is sparse symmetric positive definite. The background error covariance matrix has multiple roles in the data assimilation process. First of all, it gives an information about the accuracy of the background vector giving a statistical indication on the distance between the background vector and the true state vector. In addition, this matrix is used to spread information in spatial zones where few observations are available or to smooth information in spatial zones with a lot of observations. It is also used to solve the ill-posed problem when the number of observations is fewer than the size of the state vector.

Specifying appropriate background error covariance matrices is a complex research problem which requires careful consideration of physical, statistical and computational issues. To describe the different parts which form a background error covariance matrix, we use a simple example. Assuming that the state vector contains values of two different physical fields (salinity, with subscript S, and temperature, with subscript T), available on a three dimensional grid, the background can be defined by the following product

$$\mathbf{B} = \mathbf{K}\mathbf{D}^{1/2}\mathbf{F}\mathbf{F}^T\mathbf{D}^{1/2}\mathbf{K}^T$$

where

$$\mathbf{K} = \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{K}_{ST} & \mathbf{I} \end{pmatrix},$$

$$\mathbf{D}^{1/2} = \begin{pmatrix} \mathbf{D}_T^{1/2} & 0 \\ 0 & \mathbf{D}_S^{1/2} \end{pmatrix},$$

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}_{TT} & 0 \\ 0 & \mathbf{F}_{SS} \end{pmatrix},$$

The off-diagonal block $\mathbf{K}_{ST}$ of the matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ represents the multivariate correlations between the salinity and the temperature. Balance operators are used to model these blocks (Weaver et al., 2005). The block diagonal matrix $\mathbf{D}^{1/2} \in \mathbb{R}^{n\times n}$ contains the standard deviation of each variable of the state vector. Finally, the block diagonal matrix $\mathbf{F} \in \mathbb{R}^{n\times n}$ represents the spatial mono-variate correlations between values of the same physical field at different grid points. These blocks are modeled by integrating diffusion equations

(Weaver and Courtier, 2001) and gives the smoothing effect of the background error covariance matrix. The choice of explicit or implicit schemes for the integration of the diffusion equations depends on the desired accuracy as well as the computational budget. Implicit schemes are more robust, but require efficient linear system solvers.

For large state vectors, it is not affordable to store the matrix $\mathbf{B}$ explicitly. Moreover, the amount of information to specify each entry of the matrix is not sufficient. In practice, the matrix depends only of a small set of parameters used to define the standard deviation, the balance operators and the diffusion operators.

2.5 Inverse problem

Our focus in this entire work is to use and develop methods to solve data assimilation problems. This kind of problems are also called parameter estimation problems or inverse problems. The solution of such problems is intrinsically hard for multiple reasons:

- The existence of a solution is not guaranteed. In operational applications, the model never fits the observations because it only approximates the real behavior of the system.
- If the exact solution exists, it may not be unique. If the model operator is linear and rank-deficient, there is a nontrivial null space. Any linear combination of vectors in the null space can be added to a particular solution of the problem giving a new solution.
- Instability in the computing process often occurs. It means that the solutions of data assimilation problems are extremely unstable for small changes in the observations. This kind of problem is referred as *ill-posed* or *ill-conditioned*. The solution is to stabilize the problem and prevent overfitting with regularization techniques. To this aim, one possibility is to use the background error covariance matrix with its smoothing effect coming from the diffusion operator $\mathbf{F}$ (Weaver and Courtier, 2001).

Besides these intrinsic difficulties (see Aster et al., 2005, for details), other ones arise in the implementation of data assimilation methods. These problems will be addressed in the next chapters.

Chapter 3

The variational approach

The *variational approach* constitutes a broad class of methods which aims to solve data assimilation problems. This approach has been introduced in the early fifties by Yoshikazu Sasaki (Sasaki, 1958, Lewis and Lakshmivarahan, 2008). It treats the data assimilation problems in the form of minimizing functions measuring weighted distances between the model and the available information such as the background and the observations. It is tied up to least squares problems initially developed in the beginning of the 19th century by the German Carl Friedrich Gauss, the French Adrien-Marie Legendre and the American Robert Adrain (Lewis et al., 2006). Under the general name of variational approach, there is a set of different formulations of optimization problems which solve the data assimilation problems in different ways. The modern variational approach is divided into two general categories. The first one, labeled *3D-Var* (three-dimensional variational), has become very popular in the early nineties and operational for the first time at the National Centers for Environmental Prediction (USA) in 1991 (Parrish and Derber, 1992). The second category, labeled *4D-Var* (four-dimensional variational), is a straightforward extension of the 3D-Var one which allows to assimilate the observations at their correct observation times, see Section 3.2. The first operational 4D-Var scheme was implemented at the European Centre for Medium-Range Weather Forecasts (ECMWF) in 1996 (Courtier et al., 1994, Rabier et al., 2000a). Nowadays, other major operational centers from France, United-Kingdom, USA, Canada and Japan have adopted the 4D-Var formulation. In this chapter, we outline the principal foundations of the two categories of variational methods and present the associated formulations of the optimization problems. The methods used to solve these optimization problems are presented formally in Chapter 6. However, a first rough idea is presented in this chapter since it is useful to understand the differences between 3D-Var and 4D-Var approaches. The gap between these is bridged by the 3D-Var FGAT approach. The end of this chapter is devoted to the presentation of two reduced formulations which

allow to decrease the computational cost involved in the solution of the 4D-Var problem.

3.1 3D-Var

Assume that at time t_i, a background vector $\mathbf{x}^b$ and an observation vector $\mathbf{y}_i$ are available. A possible idea is to use both information to compute an estimation of the true state vector at time t_i. More precisely, the estimation can be computed by minimizing the distance with both information weighted by their accuracies. The accuracy of the background is represented by the inverse of the background error covariance matrix, $\mathbf{B}^{-1}$, while the accuracy of the observation vector is given by the inverse of the observation error covariance matrix, $\mathbf{R}_i^{-1}$. Mathematically, the following optimization problem is stated

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}(\mathbf{x}-\mathbf{x}^b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}^b) + \frac{1}{2}(\mathcal{H}_i\mathbf{x}-\mathbf{y}_i)^T\mathbf{R}_i^{-1}(\mathcal{H}_i\mathbf{x}-\mathbf{y}_i), \tag{3.1}$$

where the first term measures the weighted distance between the state vector and the background and the second term measures the weighted distance between the state vector and the observation vector $\mathbf{y}_i$. The solution of this problem is normally a better estimate than the background vector since it uses the information from the observation. The minimization process of (3.1) is often called the *analysis phase* while the obtained solution is called the *analysis* and denoted $\mathbf{x}^a$. Once the analysis is computed, it can be used as an initial condition to integrate the model to time t_{i+1} using the model operator $\mathcal{M}_{i+1,i}$. This integration is often called a *forecast phase*. The obtained state vector is then called the *forecast* and is usually denoted $\mathbf{x}^f$. It plays the role of the background at time t_{i+1} where a new optimization problem similar to (3.1) can be solved to assimilate the next observation vector $\mathbf{y}_{i+1}$. This process continue recursively, assimilating the observation vector one at a time. In Figure 3.1, the analysis phase is illustrated for the assimilation of a single observation vector $\mathbf{y}_i$. Some comments must be made on the formulation (3.1) and its application for operational weather forecasts. First of all, the discretization in the state vector allows us to solve (3.1) using the theory of optimization in finite dimensional vector spaces with multivariate calculus instead of using the theory of optimization in infinite dimensional spaces with calculus of variations. It is an advantage from an optimization point of view since most of the optimization algorithms developed in the last decades for engineering problems are designed for finite dimensional vector spaces. Secondly, one can see that this optimization problem belongs to the large class of nonlinear least-squares problems (remembering that the observation operator is assumed to be nonlinear). Finally and unfortunately, this process cannot be applied for operational weather forecasts and is only theoretical. Indeed, as said previously, a satellite can perform 400.000 observations in one day. It is

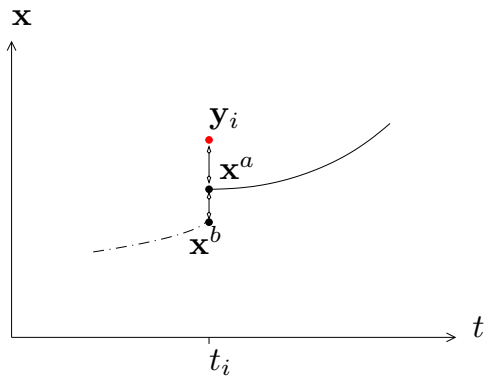

Figure 3.1 — *The analysis phase for the observation* $\mathbf{y}_i$.

not affordable to construct and minimize an optimization problem of the form of (3.1) each time an observation becomes available.

In practice, in order to reduce the number of analysis phases, a data assimilation window of appropriate size $[t_0, t_N]$ is fixed and the N observation vectors $\mathbf{y}_i$, $i = 1, \ldots, N$, available respectively at time t_i, $i = 1, \ldots, N$, are selected. All of these observations are assimilated all at once by minimizing the function

$$\min_{\mathbf{x}\in\mathbb{R}^n} J^{3\mathrm{D}}(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}^b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}^b)+\frac{1}{2}\sum_{i=1}^{N}(\mathcal{H}_i\mathbf{x}-\mathbf{y}_i)^T\mathbf{R}_i^{-1}(\mathcal{H}_i\mathbf{x}-\mathbf{y}_i). \quad (3.2)$$

This function is called the *3D-Var formulation* or simply the *3D-Var function*. The minimization of the 3D-Var formulation is called the *3D-Var problem*. The analysis obtained by solving the 3D-Var problem is assumed to be an estimation of the state of the system at the center of the data assimilation window. It can be used as an initial condition to perform a forecast for the center of the next assimilation window. This forecast plays the role of the background vector for the next data assimilation problem. The scheme is presented in Figure 3.2 and shows that using a 3D-Var formulation is equivalent to assume that all the observations in the data assimilation window have been measured at its center. This method does not take account of the observation times, except that the observations must be included in the correct data assimilation window.

A formal and classical approach from the optimization community for solving nonlinear least-squares problems such as (3.2) is given in Chapter 6. However, the meteorological community has also been interested by this kind of problem and they have developed their own optimization methods using their own vocabulary. We present this content in the following and will relate it with formal optimization methods in Chapter 6. Le Dimet and Talagrand (1986), Talagrand and Courtier (1987), Courtier and Talagrand (1987) have introduced the *incremental method* to solve variational problems such as (3.2). This is an

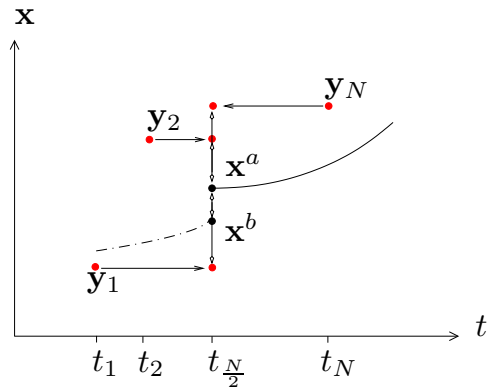

Figure 3.2 — *3D-Var scheme.*

iterative method which computes a sequence of vectors $\{\mathbf{x}_k\}$ hoping to reduce gradually the value of the 3D-Var function (3.2). Note that in the incremental formulation, the subscript does not denote the time index of the state vector but the number of the iterate. Each iterate is defined as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \delta\mathbf{x}_k,$$

where the increment $\delta\mathbf{x}_k$ is computed by solving

$$\min_{\delta\mathbf{x}\in\mathbb{R}^n} J^{\texttt{3D-Inc}}(\delta\mathbf{x}) = \frac{1}{2}(\delta\mathbf{x} - \hat{\mathbf{d}}^{bk})^T\mathbf{B}^{-1}(\delta\mathbf{x} - \hat{\mathbf{d}}^{bk}) + \frac{1}{2}\sum_{i=1}^{N}(\mathbf{H}_i^k\delta\mathbf{x} - \hat{\mathbf{d}}_i^{ok})^T\mathbf{R}_i^{-1}(\mathbf{H}_i^k\delta\mathbf{x} - \hat{\mathbf{d}}_i^{ok}). \quad (3.3)$$

The vectors $\hat{\mathbf{d}}^{bk} = \mathbf{x}^b - \mathbf{x}_k$ and $\hat{\mathbf{d}}_i^{ok} = \mathbf{y}_i - \mathcal{H}_i\mathbf{x}_k$ are called the *background departure vector* and the *observation departure vector*, respectively. The matrix $\mathbf{H}_i^k$ is the Jacobian matrix of $\mathcal{H}_i$ at x_k. The problem (3.3) is called the *incremental 3D-Var problem*. It is a local approximation of (3.2) where the observation operator has been linearized around $\mathbf{x}_k$ as

$$\mathcal{H}_i(\mathbf{x}_k + \delta\mathbf{x}) \approx \mathcal{H}_i\mathbf{x}_k + \mathbf{H}_i^k\delta\mathbf{x}.$$

It is important to remark that this approximation is suitable only for acceptable magnitude of the perturbation $\delta\mathbf{x}$ which depends on the application (Trémolet, 2004). The incremental problem is a linear least squares problem and its solution can be calculated by using linear systems solvers. It is convenient to choose the background vector as the first iterate of the sequence, i.e., $\mathbf{x}_0 = \mathbf{x}^b$, since it is the best estimate before the assimilation of the N available observations. The use of the 3D-Var approach implies the computation of the Jacobian matrix of

the nonlinear observation operator, namely $\mathbf{H}_i^k$. For operational oceanic and atmospheric models, such computation is not trivial. This problem is addressed in Chapter 7 with more details on Jacobian computation in Section 7.2.

3.2 4D-Var

The main drawback of the 3D-Var formulation is that the observations are shifted in the middle of the data assimilation window. This can potentially lead to inaccurate analysis vectors for fast moving weather systems. The 4D-Var formulation address this problem by assimilating the observations at their true observation times (Rabier and Courtier, 1992, Rabier, 2005). To this aim, the model operator $\mathcal{M}_{i+1,i}$ is used to propagate the information along the whole data assimilation window. The *4D-Var problem* is stated as

$$\min_{\mathbf{x}_0,\mathbf{x}_1,\ldots,\mathbf{x}_N\in\mathbb{R}^n} J^{\mathtt{4D}}(\mathbf{x}_0,\mathbf{x}_1,\ldots,\mathbf{x}_N) = \frac{1}{2}(\mathbf{x}_0-\mathbf{x}^b)^T\mathbf{B}^{-1}(\mathbf{x}_0-\mathbf{x}^b) + \frac{1}{2}\sum_{i=1}^{N}(\mathcal{H}_i\mathbf{x}_i-\mathbf{y}_i)^T\mathbf{R}_i^{-1}(\mathcal{H}_i\mathbf{x}_i-\mathbf{y}_i), \quad (3.4)$$

subject to the constraints

$$\mathbf{x}_{i+1} = \mathcal{M}_{i+1,i}\mathbf{x}_i \ , \quad i=0,\ldots,N-1.$$

This is an optimization problem depending on $N+1$ state vectors $\mathbf{x}_0,\mathbf{x}_1,\ldots,\mathbf{x}_N$ and constrained by N nonlinear relations. Since we have assumed that the model operator is perfect, we can directly put the constraints in the 4D-Var function, reducing the number of unknowns to the initial vector $\mathbf{x}_0$. In this case, the subscript is usually forgotten and the 4D-Var problem becomes

$$\min_{\mathbf{x}\in\mathbb{R}^n} J^{\mathtt{4D}}(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}^b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}^b) + \frac{1}{2}\sum_{i=1}^{N}(\mathcal{H}_i\mathcal{M}_{i,0}\mathbf{x}-\mathbf{y}_i)^T\mathbf{R}_i^{-1}(\mathcal{H}_i\mathcal{M}_{i,0}\mathbf{x}-\mathbf{y}_i), \quad (3.5)$$

where $\mathcal{M}_{i,0} = \prod_{j=0}^{j=i-1}\mathcal{M}_{j+1,j}$ is an operator that describes the integration of the model from time t_0 to time t_i. The scheme of the 4D-Var problem is represented in Figure 3.3. One can see that the observations are assimilated at their true observation times measuring the exact misfit between the observation vector $\mathbf{y}_i$ and the model counterpart at time t_i, given by $\mathcal{H}_i\mathcal{M}_{i,0}\mathbf{x}$. In this work, the 4D-Var formulation is presented in a *strong constraint* formulation since the constraints are used to reduce the minimization problem (3.4) to an initial value problem (3.5). When model errors exist, there is a *weak constraint* 4D-Var formulation which adds a third term accounting for the model error

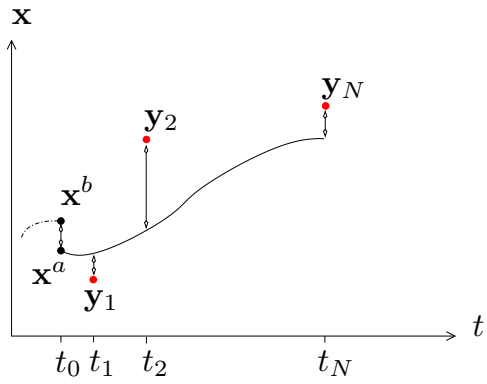

Figure 3.3 — *4D-Var scheme.*

(Trémolet, 2006). In the weak formulation, the 4D-Var function depends not only on the initial state vector but also on the model errors.

As the 4D-Var problem will be used in a large part of this work, we reformulate (3.5) in a more convenient way as

$$\min_{\mathbf{x}\in\mathbb{R}^n} J^{\mathtt{4D}}(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}^b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}^b) + \frac{1}{2}(\mathcal{G}(\mathbf{x})-\mathbf{y})^T\mathbf{R}^{-1}(\mathcal{G}(\mathbf{x})-\mathbf{y}), \quad (3.6)$$

where $\mathbf{y} = ((\mathbf{y}_1)^T, \ldots, (\mathbf{y}_N)^T)^T \in \mathbb{R}^p$, with $\mathbf{y}_i \in \mathbb{R}^{p_i}$ and $p = \sum_{i=1}^N p_i$, is the observation vector and the covariance matrix $\mathbf{R} \in \mathbb{R}^{p\times p}$ is formed with each $\mathbf{R}_i$, for $i = 1, \ldots, N$. The operator $\mathcal{G} : \mathbb{R}^n \to \mathbb{R}^p$ is mapping the initial condition into the space of the observation vector $\mathbf{y}$ and is given by

$$\mathcal{G}(\mathbf{x}) = \begin{pmatrix} \mathcal{H}_1\mathcal{M}_{1,0}\mathbf{x} \\ \vdots \\ \mathcal{H}_i\mathcal{M}_{i,0}\mathbf{x} \\ \vdots \\ \mathcal{H}_N\mathcal{M}_{N,0}\mathbf{x} \end{pmatrix}. \quad (3.7)$$

An incremental method similar to the one used for the 3D-Var problem can be developed to solve the 4D-Var problem (3.6). It leads to a sequence of *4D-Var incremental problems* defined by

$$\min_{\delta\mathbf{x}\in\mathbb{R}^n} J^{\mathtt{4D\text{-}inc}}(\delta\mathbf{x}) = (\delta\mathbf{x}-\mathbf{d}^{bk})^{\mathrm{T}}\mathbf{B}^{-1}(\delta\mathbf{x}-\mathbf{d}^{bk}) + \frac{1}{2}(\mathbf{G}_k\delta\mathbf{x}-\mathbf{d}^{ok})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{G}_k\delta\mathbf{x}-\mathbf{d}^{ok}), \quad (3.8)$$

where the matrix $\mathbf{G}_k$ is the Jacobian matrix of the operator $\mathcal{G}$ at $\mathbf{x}_k$ and is given by

$$\mathbf{G}_k = \begin{pmatrix} \mathbf{H}_1^k \mathbf{M}_{1,0}^k \\ \vdots \\ \mathbf{H}_i^k \mathbf{M}_{i,0}^k \\ \vdots \\ \mathbf{H}_N^k \mathbf{M}_{N,0}^k \end{pmatrix}, \tag{3.9}$$

where $\mathbf{M}_{i,0}^k$, $i = 1, \ldots, N$, is the Jacobian matrix of $\mathcal{M}_{i,0}$ at $\mathbf{x}_k$ and is called the tangent linear model. The background departure vector $\mathbf{d}^{bk}$ is unchanged compared to the 3D-Var incremental formulation (3.3) and is still given by

$$\mathbf{d}^{bk} = \mathbf{x}^b - \mathbf{x}_k, \tag{3.10}$$

while the observation departure vector is given now by

$$\mathbf{d}^{ok} = \mathbf{y} - \mathcal{G}(\mathbf{x}_k). \tag{3.11}$$

The incremental 4D-Var formulation (3.8) is derived using the following linearization

$$\mathcal{G}(\mathbf{x}_k + \delta\mathbf{x}) \approx \mathcal{G}(\mathbf{x}_k) + \mathbf{G}_k \delta\mathbf{x}.$$

As for the 3D-Var, the linearization is only suitable for acceptable magnitudes of perturbations. The solution of (3.6) gives an estimate of the state of the system at the initial time t_0. It is used as an initial condition to perform a forecast for the beginning of the next assimilation window.

The 4D-Var formulation outperforms the 3D-Var one since it assimilates the observations at their correct observation times (Lorenc and Rawlins, 2005). Nevertheless, the computational cost of the 4D-Var formulation is higher than the 3D-Var one since it requires the computation of the Jacobian matrices $\mathbf{H}_i^k$ and $\mathbf{M}_i^k$, while the 3D-Var formulation requires only the computation of $\mathbf{H}_i^k$. The problem of Jacobian computation is addressed in Chapter 7 with more details in Section 7.2.

3.3 Other formulations

The 3D-Var and 4D-Var formulations are not the only possible formulations to solve data assimilation problems. In this section, we shortly present two other famous formulations. The first one bridges the gap between the 3D-Var and 4D-Var formulations and is called the 3D-FGAT which stands for First Guess at Appropriate Time (Massart et al., 2010). It uses the same formulation as the 4D-Var formulation (3.6). The only difference is in the incremental

formulation where the Jacobian of the model operator is assumed to be the identity. Mathematically, the following linearization is used in the 3D-FGAT

$$\mathcal{G}(\mathbf{x}_k + \delta\mathbf{x}) \approx \mathcal{G}(\mathbf{x}_k) + \mathbf{H}^k \delta\mathbf{x},$$

where $\mathbf{H}^k$ is the Jacobian matrix of

$$\mathcal{H}(\mathbf{x}) = \begin{pmatrix} \mathcal{H}_1\mathbf{x} \\ \vdots \\ \mathcal{H}_i\mathbf{x} \\ \vdots \\ \mathcal{H}_N\mathbf{x} \end{pmatrix}$$

at $\mathbf{x}_k$. It leads to the following incremental problem

$$\min_{\delta\mathbf{x}\in\mathbb{R}^n} J^{\texttt{3D-FGAT-Inc}}(\delta\mathbf{x}) = (\delta\mathbf{x} - \mathbf{d}^{bk})^{\mathrm{T}}\mathbf{B}^{-1}(\delta\mathbf{x} - \mathbf{d}^{bk}) + \frac{1}{2}(\mathbf{H}^k\delta\mathbf{x} - \mathbf{d}^{ok})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{H}^k\delta\mathbf{x} - \mathbf{d}^{ok}). \quad (3.12)$$

The 3D-FGAT method loses the dynamical aspect of the 4D-Var since the increment does not evolve dynamically in the assimilation window. It is for that reason that it belongs to the 3D-Var category. Nevertheless, the comparison between the model and the observations is still computed at the right observation time in the departure vector. It is an attractive compromise between the accuracy of the 4D-Var formulation and the simplicity of the 3D-Var formulation which requires only the computation of the Jacobian matrices of the observation operators.

A last famous formulation to solve data assimilation problems is called PSAS which stands for Physical Space Assimilation System (Courtier, 1997). This approach is available for both 3D-Var and 4D-Var categories. It is a dual method which solves the incremental problem in the observation space rather than in the state space. It can be an attractive formulation since the observation space is often substantially smaller than the state space. Theoretically, the PSAS formulation gives the same solution than the conventional variational formulations.

3.4 Reduced order 4D-Var

The 4D-Var approach (3.6) with its incremental formulation (3.8) can be computationally intensive for operational problems. Indeed, the 4D-Var function is a very large nonlinear least squares function which is expensive to evaluate since it involves a model integration along the data assimilation window.

Moreover, the computation of the Jacobian matrix $\mathbf{G}_k$ used in the incremental formulation is also expensive. Many attempts have been made to reduce the computational cost of the 4D-Var approach. We briefly review some of the most famous ones.

A first possible idea is to minimize the sequence of 4D-Var incremental functions (3.8) using a coarser discretization grid than the one used for the computation of the departure vectors (Courtier et al., 1994). It leads to a computation gain since the minimization is performed in a much smaller space, say $\mathbb{R}^r$ with $r << n$. More precisely, a reduced increment $\delta\underline{\mathbf{x}} \in \mathbb{R}^r$ is computed at low resolution on a coarse grid and the following update formula

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{S}\delta\underline{\mathbf{x}}_k,$$

is used where $\delta\underline{\mathbf{x}}$ is prolongated at high resolution on the fine grid using a linear prolongation operator $\mathbf{S} \in \mathbb{R}^{n\times r}$. Conversely, the operator which transforms a vector from high to low resolution is a restriction operator and is given by $\mathbf{S}^+$, the pseudo-inverse of $\mathbf{S}$ (see Appendix A.1). With these prolongation and restriction operators, the *reduced incremental 4D-Var problem* can be stated as

$$\min_{\delta\underline{\mathbf{x}}\in\mathbb{R}^r} (\delta\underline{\mathbf{x}} - \mathbf{S}^+\mathbf{d}^{bk})^{\mathrm{T}}\mathbf{B}_r^{-1}(\delta\underline{\mathbf{x}} - \mathbf{S}^+\mathbf{d}^{bk}) + \frac{1}{2}(\tilde{\mathbf{G}}_k\delta\underline{\mathbf{x}} - \mathbf{d}^{ok})^{\mathrm{T}}\mathbf{R}^{-1}(\tilde{\mathbf{G}}_k\delta\underline{\mathbf{x}} - \mathbf{d}^{ok}), \quad (3.13)$$

where $\mathbf{B}_r \in \mathbb{R}^{r\times r}$ is the reduced background error covariance matrix. It is an approximation in some sense of $\mathbf{S}^+\mathbf{B}\mathbf{S}$ and models the background error on the coarse grid. Its inverse $\mathbf{B}_r^{-1}$ is an approximation of $\mathbf{S}^+\mathbf{B}^{-1}\mathbf{S}$. The matrix $\tilde{\mathbf{G}}_k$ is the Jacobian matrix of $\tilde{\mathcal{G}} : \mathbb{R}^r \to \mathbb{R}^p$ which maps an initial condition defined on the coarse grid, $\underline{\mathbf{x}} \in \mathbb{R}^r$, into the space of the observation vectors. It is defined by

$$\tilde{\mathcal{G}}(\underline{\mathbf{x}}) = \begin{pmatrix} \tilde{\mathcal{H}}_1\tilde{\mathcal{M}}_{1,0}\underline{\mathbf{x}} \\ \vdots \\ \tilde{\mathcal{H}}_i\tilde{\mathcal{M}}_{i,0}\underline{\mathbf{x}} \\ \vdots \\ \tilde{\mathcal{H}}_N\tilde{\mathcal{M}}_{N,0}\underline{\mathbf{x}} \end{pmatrix},$$

where $\tilde{\mathcal{M}}_{i,0}$, $i = 1, \ldots, N$, is the coarse model operator, from time t_0 to time t_i, given by

$$\tilde{\mathcal{M}}_{i,0} : \mathbb{R}^r \to \mathbb{R}^r : \underline{\mathbf{x}} \rightsquigarrow \tilde{\mathcal{M}}_{i,0}\underline{\mathbf{x}},$$

and where $\tilde{\mathcal{H}}_i$, $i = 1, \ldots, N$, is the coarse observation operator given by

$$\tilde{\mathcal{H}}_i : \mathbb{R}^r \to \mathbb{R}^{p_i} : \underline{\mathbf{x}} \rightsquigarrow \tilde{\mathcal{H}}_i\underline{\mathbf{x}}.$$

These two operators defined on the coarse grid must approximate in some sense $\mathbf{S}^+\mathcal{M}_{i,0}\mathbf{S}$ and $\mathcal{H}_i\mathbf{S}$, respectively. As said previously, one can remark that the

departure vectors of (3.13), defined by (3.10) and (3.11), are computed on the fine grid. This procedure of reduced incremental 4D-Var can be slightly modified. Instead of only using one coarse grid, a hierarchy of different grids (from the coarsest to the finest) can be used to solve the sequence of incremental 4D-Var problems. This strategy of refinement is called the *multi-incremental 4D-Var* (Veerse and Thépaut, 1998, Laroche and Gauthier, 1998). It is possible to use the same refinement philosophy to derive another reduced formulation. Rather than using a hierarchy of discretization grids, one can use a hierarchy of model operators which simplify the physical assumptions and discard some physical process (Rabier et al., 2000b, Mahfouf and Rabier, 2000).

Besides these ideas, it is also possible to formulate another reduced incremental 4D-Var problem which performs the minimization of the classical 4D-Var incremental function (3.8) only in some well-chosen directions. With this philosophy, another reduced increment $\delta\underline{\mathbf{x}}_k \in \mathbb{R}^r$ is computed by solving

$$\min_{\delta\underline{\mathbf{x}}\in\mathbb{R}^r} (\mathbf{L}\delta\underline{\mathbf{x}} - \mathbf{d}^{bk})^{\mathrm{T}}\mathbf{B}^{-1}(\mathbf{L}\delta\underline{\mathbf{x}} - \mathbf{d}^{bk}) + \frac{1}{2}(\mathbf{G}_k\mathbf{L}\delta\underline{\mathbf{x}} - \mathbf{d}^{ok})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{G}_k\mathbf{L}\delta\underline{\mathbf{x}} - \mathbf{d}^{ok}), \quad (3.14)$$

where the matrix $\mathbf{L} \in \mathbb{R}^{n\times r}$ contains the r favored directions in which the optimization is performed. The update formula is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{L}\delta\underline{\mathbf{x}}_k.$$

The choice of the favored directions can be performed using different approaches such as the balance truncation (Lawless et al., 2006) or based on empirical orthogonal functions (Robert et al., 2005). These last ones will be used to establish, for linear operators, a connection between a reduced 4D-Var problem and a SEEK filter derived from the sequential approach (see Theorem 5.3). Moreover, the reduced incremental 4D-Var formulation (3.14) will be used in Section 6.4 to compute an appropriate starting point for the incremental method .

3.5 Evolution of the background error

We have seen in Sections 3.1 and 3.2 that the variational approach computes an analysis vector which is an estimation of the system state at a given time. This analysis can be used as an initial condition to perform a forecast and to compute the background of the next assimilation window. However, the variational approach does not explicitly calculate the evolution of the background error covariance matrix. Since the background vector evolves dynamically during the sequence of assimilation problems, it seems natural that its error covariance matrix does the same. A possible idea to estimate the background error statistics is to solve an ensemble of independent variational problems where the

observations have been perturbed by adding well-chosen random noises from the assumed distribution of the observation error (Daget et al., 2009, Bonavita et al., 2012). Each variational problem produces its own sequence of analysis and background vectors. The difference between the background vectors for pairs of members of the ensemble can be used to estimate the background error covariance matrix. With this technique, the background error covariance matrix evolves dynamically along the sequence of data assimilation windows.

Chapter 4

The sequential approach

The *sequential approach* is the second broad class of methods to solve data assimilation problems. This approach is based on statistical estimation theory and has been introduced by Rudolph Kalman (1960) with the *Kalman filter*. This filter has been successfully applied in a wide range of engineering applications, mostly in navigation and positioning systems. However, its application in operational weather forecasting is not straightforward. Some adaptations of the original filter have to be performed to overcome the difficulties arising from the large size of the problems and from the nonlinearities in the model and observation operators. It has led to the development of some suboptimal filters (see Todling and Cohn, 1994, Rozier et al., 2007, for a review). Among them, the *Singular Evolutive Extended Kalman* (SEEK) filter decreases the computational cost by reducing the rank of the estimator error covariance matrices (Pham et al., 1998). This filter is of interest since it has been successfully applied to operational oceanographic forecast (Hoteit and Pham, 2003).

The term "filter" has two meanings in our work. The first one comes from the usual and general definition which states a filter as *a porous material through which a liquid or gas is passed in order to separate the fluid from suspended particulate matter*. This definition, applied in our context, means that the filter is a mathematical device through which the system's information is passed to separate the information from its errors. The second meaning of the term filter is only related to the context of data assimilation problems. It is used to denote a process which estimates the state of a system at a given time t_i using only observations up to this time, i.e., $\{\mathbf{y}_1, \ldots, \mathbf{y}_i\}$. In conjunction with this term, two other terms are usually used. A "smoother" estimates the state of a system at a given time t_i using observation up to but also posterior to this time, i.e., $\{\mathbf{y}_1, \ldots, \mathbf{y}_i, \mathbf{y}_{i+1}, \ldots, \mathbf{y}_{i+L}\}$, where the constant L is called the lag of the smoother. Whereas, a "predictor" estimates the state of a system at a given time t_i using only some observations prior to this time, i.e., $\{\mathbf{y}_1, \ldots, \mathbf{y}_{i-F}\}$, where the constant F defines the length of the prediction.

In our work, we are only interested by filter processing (see Simon, 2006, for details on smoothers and predictors).

In this chapter, we derive the equations of the Kalman filter using a non-conventional approach based on the theory of constrained optimization. Once the equations are established, we show why these cannot be applied directly for large size problems by performing a count of the required number of floating-point operations (flops). The SEEK filter which approximates the original Kalman filter is then presented. A flops analysis is fulfilled showing the ability of the SEEK filter to handle large size problems. Finally, we describe how the SEEK filter can be applied in practice. For the whole chapter, we assume that the model and observation operators are linear. It is a legitimate approach since the Kalman filter has been originally introduced for linear operators. The problems arising from nonlinearities is not covered in this chapter but is treated in Chapter 7.

4.1 Estimation

The sequential approach with its Kalman filter has been established using a rather different philosophy than the variational approach. The true trajectory of a system during a time interval is represented by a sequence of true state vectors. Since these vectors are unknown, they can be represented using random vectors. The idea of the Kalman filter is to produce a sequence of *estimators* which approximates the sequence of the true state vectors. These estimators are constructed using the information contained in the available observations and in the dynamics of the model operator.

Moreover, the Kalman filter computes a sequence of *estimator error covariance matrices* giving statistical information about the accuracy of the estimators. Since the numerical integration of the system's equations implies a space and time discretization, this chapter will deal only with discrete Kalman filters (see Simon, 2006, for continuous Kalman filters). To derive the equations of the Kalman filter, a progressive approach based on the theory of constrained optimization will be used. Although this is not the shortest approach, we think that it gives a profound understanding of the theory behind the Kalman filter. We first explain a method to produce an optimal estimator of a state vector using one observation. Then, we present a recursive estimation scheme which allows to update an estimator when new observations become available. Finally, we describe how to propagate estimators and error covariance matrices in time. The Kalman filter equations will be naturally derived gathering all these ingredients.

4.1.1 Statistical estimation

In this part, we show how to produce the *best linear unbiased estimator* (BLUE) of a true state vector $\mathbf{x}_i^t$ available at time t_i using the information contained in one observation $\mathbf{y}_i = \mathbf{H}_i\mathbf{x}_i^t + \boldsymbol{\epsilon}_i^o$. The BLUE estimator is denoted $\mathbf{x}_i$ and the optimality is ensured by minimizing its variance. As assumed in Chapter 2, the mean of the observation error, $E\left[\boldsymbol{\epsilon}_i^o\right]$, is zeros and the observation error covariance matrix, $E\left[\boldsymbol{\epsilon}_i^o(\boldsymbol{\epsilon}_i^o)^T\right]$, is given by $\mathbf{R}_i$. The matrix $\mathbf{R}_i$ is symmetric positive definite by construction (see Subsection 2.3.3). The desired estimator must be linear in the observation and can thus be written has

$$\mathbf{x}_i = \mathbf{T}\mathbf{y}_i, \tag{4.1}$$

where $\mathbf{T} \in \mathbb{R}^{n\times p_i}$ has to be determined. The error between the estimator and the true state vector, denoted $\boldsymbol{\epsilon}_i^x$, can be expressed as

$$\begin{aligned}\boldsymbol{\epsilon}_i^x &= \mathbf{x}_i - \mathbf{x}_i^t \\ &= \mathbf{T}\mathbf{y}_i - \mathbf{x}_i^t \\ &= \mathbf{T}\left(\mathbf{H}_i\mathbf{x}_i^t + \boldsymbol{\epsilon}_i^o\right) - \mathbf{x}_i^t \\ &= \left(\mathbf{T}\mathbf{H}_i - \mathbf{I}_n\right)\mathbf{x}_i^t + \mathbf{T}\boldsymbol{\epsilon}_i^o,\end{aligned} \tag{4.2}$$

using (4.1) and the definition of $\mathbf{y}_i$. Note that the matrix $\mathbf{I}_n$ is the identity matrix of order n. The mean of this estimator error is given by

$$\begin{aligned}E[\boldsymbol{\epsilon}_i^x] &= E\left[(\mathbf{T}\mathbf{H}_i - \mathbf{I}_n)\mathbf{x}_i^t + \mathbf{T}\boldsymbol{\epsilon}_i^o\right] \\ &= \left(\mathbf{T}\mathbf{H}_i - \mathbf{I}_n\right)E\left[\mathbf{x}_i^t\right] + \mathbf{T}E\left[\boldsymbol{\epsilon}_i^o\right] \\ &= \left(\mathbf{T}\mathbf{H}_i - \mathbf{I}_n\right)E\left[\mathbf{x}_i^t\right],\end{aligned}$$

using the properties of the mean and the assumption that $E[\boldsymbol{\epsilon}_i^o] = 0$. The condition of an unbiased estimator imposes that the mean of the estimator error is equal to zero, i.e., $E[\boldsymbol{\epsilon}_i^x] = 0$. It implies that we have to impose the condition $\mathbf{T}\mathbf{H}_i - \mathbf{I}_n = 0$, since $E[\mathbf{x}_i^t] \neq 0$ in general. Under this condition, the estimator error (4.2) becomes

$$\boldsymbol{\epsilon}_i^x = \mathbf{T}\boldsymbol{\epsilon}_i^o$$

and the estimator error covariance matrix, denoted $\mathbf{P}_i$, is given by

$$\begin{aligned}\mathbf{P}_i =& E\left[\boldsymbol{\epsilon}_i^x\left(\boldsymbol{\epsilon}_i^x\right)^T\right] \\ =& E\left[\mathbf{T}\boldsymbol{\epsilon}_i^o\left(\mathbf{T}\boldsymbol{\epsilon}_i^o\right)^T\right] \\ =& \mathbf{T}E\left[\boldsymbol{\epsilon}_i^o\left(\boldsymbol{\epsilon}_i^o\right)^T\right]\mathbf{T}^T \\ =& \mathbf{T}\mathbf{R}_i\mathbf{T}^T,\end{aligned} \tag{4.3}$$

using the properties of the mean and the definition of $\mathbf{R}_i$. The variance of the estimator $\mathbf{x}_i$ is given by the trace of $\mathbf{P}_i$. To obtain the BLUE estimator, the trace of $\mathbf{P}_i$ is minimized subject to the unbiased constraint $\mathbf{TH}_i - \mathbf{I}_n = 0$. It leads to the following constrained optimization problem

$$\begin{cases} \min\limits_{\mathbf{T}\in\mathbb{R}^{n\times p_i}} tr\left(\mathbf{TR}_i\mathbf{T}^T\right) \\ s.t.\ \mathbf{TH}_i - \mathbf{I}_n = 0. \end{cases} \tag{4.4}$$

The solution can be computed using the theorem of second-order sufficient optimality conditions where the Lagrangian function is given by

$$\mathcal{L}(\mathbf{T},\mathbf{\Lambda}) = tr\left(\mathbf{TR}_i\mathbf{T}^T\right) - \sum_{k=1}^{n}\sum_{l=1}^{n}\mathbf{\Lambda}(k,l)\left[\sum_{m=1}^{p_i}\mathbf{T}(k,m)\mathbf{H}_i(m,l) - \mathbf{I}(k,l)\right],$$

with $\mathbf{\Lambda}\in\mathbb{R}^{n\times n}$, the Lagrangian multiplier matrix (see Appendix B with its last example for more details). Note that the notation (i,j) designates the matrix element at the ith row and the jth column. It gives the following Karush–Kuhn–Tucker (KKT) linear system

$$\begin{aligned} 2\mathbf{TR}_i + \mathbf{\Lambda}\mathbf{H}_i^{\mathbf{T}} &= 0 \\ \mathbf{TH}_i - \mathbf{I}_n &= 0 \end{aligned}$$

using a similar approach as for (B.7) and observing that $\nabla_{\mathbf{T}} tr\left(\mathbf{TR}_i\mathbf{T}^T\right) = 2\mathbf{TR}_i$ from (A.9).

The solution of the system is obtained by solving the first equation for $\mathbf{T}$ and by performing the substitution in the second one. It gives

$$\mathbf{T} = \left(\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i\right)^{-1}\mathbf{H}_i^T\mathbf{R}_i^{-1} \tag{4.5}$$

and

$$\mathbf{\Lambda} = -2\left(\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i\right)^{-1}.$$

where the inverse of $\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i$ exists for a full rank matrix $\mathbf{H}_i$ with $p_i \geqslant n$[1]. Otherwise, the estimation problem is ill-posed. This solution is a strict minimum of (4.4) since from (A.9) and (A.10), we have

$$\nabla^2_{\mathbf{T}}\mathcal{L}(\mathbf{T},\mathbf{\Lambda}) = 2\mathbf{R}_i$$

which is symmetric positive definite by definition. Using (4.1) and (4.5), the BLUE estimator of the vector $\mathbf{x}_i^t$ using the observation $\mathbf{y}_i$ is

$$\mathbf{x}_i = (\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i)^{-1}\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{y}_i. \tag{4.6}$$

[1] In operational weather forecasts, this approach is impractical since the number of observation, p_i, is usually much smaller than the size of the state vector, n. The solution is given in the next section with the use of a regularization term

Its error covariance matrix is given by

$$\begin{aligned}\mathbf{P}_i =& \mathbf{T}\mathbf{R}_i\mathbf{T}^T \\ &= \left(\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i\right)^{-1}\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{R}_i\left(\left(\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i\right)^{-1}\mathbf{H}_i^T\mathbf{R}_i^{-1}\right)^T \\ &= \left(\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i\right)^{-1}\mathbf{H}_i^T\left(\mathbf{R}_i^{-1}\mathbf{H}_i\left(\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i\right)^{-1}\right) \\ &= \left(\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i\right)^{-1}, \end{aligned} \tag{4.7}$$

using (4.3), (4.5) and some basic linear algebra calculus.

4.1.2 Recursive estimation

Whenever new measurements become available at time t_i, it would be inefficient to augment the vector $\mathbf{y}_i$ and the matrix $\mathbf{H}_i$ with the new measurements and to completely recompute the estimator $\mathbf{x}_i$ using (4.6) and its error covariance matrix using (4.7). It is preferable to develop a technique which updates the current estimator $\mathbf{x}_i$ with its error covariance matrix $\mathbf{P}_i$ using the new measurements. This is the aim of this part which derives a recursive process to construct a new BLUE estimator, $\mathbf{x}_i^+$, from a previous one, $\mathbf{x}_i$, and from a new available observation

$$\mathbf{y}_i^+ = \mathbf{H}_i^+\mathbf{x}_i^t + \boldsymbol{\epsilon}_i^{o+}.$$

As previously, we assume that the mean of the observation error, $E[\boldsymbol{\epsilon}_i^{o+}]$, is zero and that the observation error covariance matrix, $E[\boldsymbol{\epsilon}_i^{o+}(\boldsymbol{\epsilon}_i^{o+})^T]$, is given by $\mathbf{R}_i^+$ which is symmetric positive definite. Moreover, we assume that the estimator error of $\mathbf{x}_i$ and the observation error of $\mathbf{y}_i^+$ are uncorrelated, i.e., $E[\boldsymbol{\epsilon}_i^x(\boldsymbol{\epsilon}_i^{o+})^T] = 0$. To derive the formula which update the BLUE estimator $\mathbf{x}_i$, we use an approach similar to the previous subsection.

The estimator $\mathbf{x}_i^+ \in \mathbb{R}^n$ of the unknown vector $\mathbf{x}_i^t$ is build so as to depend linearly on the new observation $\mathbf{y}_i^+ \in \mathbb{R}^{p_i}$ and on the previous estimator $\mathbf{x}_i$. Thus, we have

$$\mathbf{x}_i^+ = \mathbf{F}\mathbf{x}_i + \mathbf{K}\mathbf{y}_i^+, \tag{4.8}$$

where $\mathbf{F} \in \mathbb{R}^{n\times n}$ and $\mathbf{K} \in \mathbb{R}^{n\times p_i}$ have to be determined. The new estimator error is given by

$$\begin{aligned}\boldsymbol{\epsilon}_i^{x+} &= \mathbf{x}_i^+ - \mathbf{x}_i^t \\ &= \mathbf{F}\mathbf{x}_i + \mathbf{K}\left(\mathbf{H}_i^+\mathbf{x}_i^t + \boldsymbol{\epsilon}_i^{o+}\right) - \mathbf{x}_i^t \\ &= \mathbf{F}\left(\mathbf{x}_i - \mathbf{x}_i^t + \mathbf{x}_i^t\right) + \mathbf{K}\left(\mathbf{H}_i^+\mathbf{x}_i^t + \boldsymbol{\epsilon}_i^{o+}\right) - \mathbf{x}_i^t \\ &= \mathbf{F}\left(\mathbf{x}_i - \mathbf{x}_i^t\right) + \mathbf{K}\boldsymbol{\epsilon}_i^{o+} + \left(\mathbf{F} + \mathbf{K}\mathbf{H}_i^+ - \mathbf{I}_n\right)\mathbf{x}_i^t \\ &= \mathbf{F}\boldsymbol{\epsilon}_i^x + \mathbf{K}\boldsymbol{\epsilon}_i^{o+} + \left(\mathbf{F} + \mathbf{K}\mathbf{H}_i^+ - \mathbf{I}_n\right)\mathbf{x}_i^t, \end{aligned} \tag{4.9}$$

and its mean is equal to

$$\begin{aligned} E[\boldsymbol{\epsilon}_i^{x+}] &= E\left[\mathbf{F}\boldsymbol{\epsilon}_i^x + \mathbf{K}\boldsymbol{\epsilon}_i^{o+} + \left(\mathbf{F} + \mathbf{K}\mathbf{H}_i^+ - \mathbf{I}_n\right)\mathbf{x}_i^t\right] \\ &= \mathbf{F}E\left[\boldsymbol{\epsilon}_i^x\right] + \mathbf{K}E\left[\boldsymbol{\epsilon}_i^{o+}\right] + \left(\mathbf{F} + \mathbf{K}\mathbf{H}_i^+ - \mathbf{I}_n\right)E\left[\mathbf{x}_i^t\right] \\ &= \left(\mathbf{F} + \mathbf{K}\mathbf{H}_i^+ - \mathbf{I}_n\right)E\left[\mathbf{x}_i^t\right], \end{aligned}$$

since $E[\boldsymbol{\epsilon}_i^x] = 0$ and $E[\boldsymbol{\epsilon}_i^{o+}] = 0$ by assumption. The estimator has to be unbiased meaning that the mean of its error has to be zero. This condition imposes that

$$\mathbf{F} + \mathbf{K}\mathbf{H}_i^+ - \mathbf{I}_n = 0, \tag{4.10}$$

since in general $E[\mathbf{x}_i^t] \neq 0$. Using the condition (4.10), the estimator error (4.9) becomes

$$\boldsymbol{\epsilon}_i^{x+} = \mathbf{F}\boldsymbol{\epsilon}_i^x + \mathbf{K}\boldsymbol{\epsilon}_i^{o+},$$

and the estimator error covariance matrix, denoted $\mathbf{P}_i^+$, is given by

$$\begin{aligned} \mathbf{P}_i^+ &= E\left[\boldsymbol{\epsilon}_i^{x+}\left(\boldsymbol{\epsilon}_i^{x+}\right)^T\right] \\ &= E\left[\left(\mathbf{F}\boldsymbol{\epsilon}_i^x + \mathbf{K}\boldsymbol{\epsilon}_i^{o+}\right)\left(\mathbf{F}\boldsymbol{\epsilon}_i^x + \mathbf{K}\boldsymbol{\epsilon}_i^{o+}\right)^T\right] \\ &= \mathbf{F}E\left[\boldsymbol{\epsilon}_i^x\left(\boldsymbol{\epsilon}_i^x\right)^T\right]\mathbf{F}^T + \mathbf{K}E\left[\boldsymbol{\epsilon}_i^{o+}\left(\boldsymbol{\epsilon}_i^{o+}\right)^T\right]\mathbf{K}^T \\ &= \mathbf{F}\mathbf{P}_i\mathbf{F}^T + \mathbf{K}\mathbf{R}_i^+\mathbf{K}^T, \end{aligned} \tag{4.11}$$

where the assumption $E[\boldsymbol{\epsilon}_i^x(\boldsymbol{\epsilon}_i^{o+})^T] = 0$ is used to simplify the third expression. To construct the new BLUE estimator, the trace of $\mathbf{P}_i^+$ is minimized subject to the unbiased constraint $\mathbf{F} + \mathbf{K}\mathbf{H}_i^+ - \mathbf{I}_n = 0$. It gives the following optimization problem

$$\begin{cases} \min\limits_{\mathbf{F}\in\mathbb{R}^{n\times n}, \mathbf{K}\in\mathbb{R}^{n\times p_i}} tr\left(\mathbf{F}\mathbf{P}_i\mathbf{F}^T + \mathbf{K}\mathbf{R}_i^+\mathbf{K}^T\right) \\ s.t.\ \mathbf{F} + \mathbf{K}\mathbf{H}_i^+ - \mathbf{I}_n = 0. \end{cases} \tag{4.12}$$

Since the objective function depends on two matrices $\mathbf{F}$ and $\mathbf{K}$, the Lagrangian function for this problem is given by

$$\begin{aligned} \mathcal{L}(\mathbf{F}, \mathbf{K}, \boldsymbol{\Lambda}) &= tr\left(\mathbf{F}\mathbf{P}_i\mathbf{F}^T + \mathbf{K}\mathbf{R}_i^+\mathbf{K}^T\right) \\ &\quad - \sum_{k=1}^{n}\sum_{l=1}^{n}\boldsymbol{\Lambda}(k,l)\left[\mathbf{F}(k,l) + \sum_{m=1}^{p_i}\mathbf{K}(k,m)\mathbf{H}_i^+(m,l) - \mathbf{I}(k,l)\right], \end{aligned}$$

where $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times n}$ is the Lagrangian multiplier matrix (see Appendix B with its last example for more details). Using the linearity of the trace and simple calculus, the Lagrangian function can be reformulated as

$$\begin{aligned} \mathcal{L}(\mathbf{F}, \mathbf{K}, \boldsymbol{\Lambda}) &= tr\left(\mathbf{F}\mathbf{P}_i\mathbf{F}^T\right) + tr\left(\mathbf{K}\mathbf{R}_i^+\mathbf{K}^T\right) \\ &\quad - \sum_{k=1}^{n}\sum_{l=1}^{n}\boldsymbol{\Lambda}(k,l)\mathbf{F}(k,l) - \sum_{l=1}^{n}\boldsymbol{\Lambda}(:,l)^T\left(\mathbf{K}\mathbf{H}_i^+(:,l) - \mathbf{I}(:,l)\right), \end{aligned}$$

where $(:, l)$ denotes the lth column of a matrix. The KKT system is formed by differentiating the Lagrangian function with respect to $\mathbf{F}$ and $\mathbf{K}$ and by adding the constraint. It gives

$$\begin{aligned} 2\mathbf{F}\mathbf{P}_i + \mathbf{\Lambda} &= 0 \\ 2\mathbf{K}\mathbf{R}_i^+ + \mathbf{\Lambda}\left(\mathbf{H}_i^+\right)^T &= 0 \\ \mathbf{F} + \mathbf{K}\mathbf{H}_i^+ - \mathbf{I}_n &= 0, \end{aligned}$$

using a similar approach as for (B.7) together with the formulas (A.9). The solution is given by

$$\begin{aligned} \mathbf{F} &= \mathbf{I}_n - \mathbf{P}_i(\mathbf{H}_i^+)^T\left(\mathbf{H}_i^+\mathbf{P}_i\left(\mathbf{H}_i^+\right)^T + \mathbf{R}_i^+\right)^{-1}\mathbf{H}_i^+ \\ \mathbf{K} &= \mathbf{P}_i(\mathbf{H}_i^+)^T\left(\mathbf{H}_i^+\mathbf{P}_i\left(\mathbf{H}_i^+\right)^T + \mathbf{R}_i^+\right)^{-1} \\ \mathbf{\Lambda} &= -2\left(\mathbf{I}_n - \mathbf{P}_i(\mathbf{H}_i^+)^T\left(\mathbf{H}_i^+\mathbf{P}_i(\mathbf{H}_i^+)^T + \mathbf{R}_i^+\right)^{-1}\mathbf{H}_i^+\right)\mathbf{P}_i. \end{aligned} \tag{4.13}$$

Note that the matrix $\mathbf{R}_i^+ + \mathbf{H}_i^+\mathbf{P}_i(\mathbf{H}_i^+)^T$ is invertible since $\mathbf{R}_i^+$ and $\mathbf{P}_i$ are positive definite matrices and $\mathbf{H}_i^+$ is assumed to be of full rank (see Section 2.3). Using (A.9) and (A.10), we have

$$\nabla^2\mathcal{L}(\mathbf{F}, \mathbf{K}, \boldsymbol{\lambda}) = \begin{pmatrix} \nabla^2_{FF}\mathcal{L} & \nabla^2_{FK}\mathcal{L} \\ \nabla^2_{KF}\mathcal{L} & \nabla^2_{KK}\mathcal{L} \end{pmatrix} = \begin{pmatrix} 2\mathbf{P}_i & 0 \\ 0 & 2\mathbf{R}_i^+ \end{pmatrix},$$

which is a symmetric positive definite matrix since $\mathbf{P}_i$ and $\mathbf{R}_i^+$ are symmetric positive definite matrices. So, one can conclude that the solution of the linear system is a strict local minimum of (4.12). Recalling the form of the estimator (4.8) and the unbiased constraint (4.10), the BLUE estimator of the vector $\mathbf{x}_i^t$ using the former estimator $\mathbf{x}_i$ and the observation $\mathbf{y}_i^+$ is then given by

$$\begin{aligned} \mathbf{x}_i^+ &= \mathbf{F}\mathbf{x}_i + \mathbf{K}\mathbf{y}_i^+ \\ &= \left(\mathbf{I}_n - \mathbf{K}\mathbf{H}_i^+\right)\mathbf{x}_i + \mathbf{K}\mathbf{y}_i^+ \\ &= \mathbf{x}_i + \mathbf{K}\left(\mathbf{y}_i^+ - \mathbf{H}_i^+\mathbf{x}_i\right) \\ &= \mathbf{x}_i + \mathbf{P}_i\left(\mathbf{H}_i^+\right)^T\left(\mathbf{H}_i^+\mathbf{P}_i\left(\mathbf{H}_i^+\right)^T + \mathbf{R}_i^+\right)^{-1}\left(\mathbf{y}_i^+ - \mathbf{H}_i^+\mathbf{x}_i\right). \end{aligned} \tag{4.14}$$

Its error covariance matrix, $\mathbf{P}_i^+$, can be written as

$$\begin{aligned} \mathbf{P}_i^+ =& \mathbf{F}\mathbf{P}_i\mathbf{F}^T + \mathbf{K}\mathbf{R}_i^+\mathbf{K}^T \\ &= \left(\mathbf{I}_n - \mathbf{K}\mathbf{H}_i^+\right)\mathbf{P}_i\left(\mathbf{I}_n - \mathbf{K}\mathbf{H}_i^+\right)^T + \mathbf{K}\mathbf{R}_i^+\mathbf{K}^T \\ &= \left(\mathbf{I}_n - \mathbf{K}\mathbf{H}_i^+\right)\left(\mathbf{P}_i - \mathbf{P}_i\left(\mathbf{H}_i^+\right)^T\mathbf{K}^T\right) + \mathbf{K}\mathbf{R}_i^+\mathbf{K}^T \\ &= \left(\mathbf{I}_n - \mathbf{K}\mathbf{H}_i^+\right)\mathbf{P}_i - \left(\mathbf{I}_n - \mathbf{K}\mathbf{H}_i^+\right)\mathbf{P}_i\left(\mathbf{H}_i^+\right)^T\mathbf{K}^T + \mathbf{K}\mathbf{R}_i^+\mathbf{K}^T \\ &= \left(\mathbf{I}_n - \mathbf{K}\mathbf{H}_i^+\right)\mathbf{P}_i + \left(\mathbf{K}\left(\mathbf{H}_i^+\mathbf{P}_i\left(\mathbf{H}_i^+\right)^T + \mathbf{R}_i^+\right) - \mathbf{P}_i\left(\mathbf{H}_i^+\right)^T\right)\mathbf{K}^T. \end{aligned}$$

This expression can be reformulated by replacing the first occurrence of $\mathbf{K}$ by its optimal value (4.13). We obtain

$$\begin{aligned}\mathbf{P}_i^+ &= \left(\mathbf{I}_n - \mathbf{K}\mathbf{H}_i^+\right)\mathbf{P}_i \\ &+ \left(\mathbf{P}(\mathbf{H}_i^+)^T\left(\mathbf{H}_i^+\mathbf{P}_i(\mathbf{H}_i^+)^T + \mathbf{R}_i^+\right)^{-1}\left(\mathbf{H}_i^+\mathbf{P}_i(\mathbf{H}_i^+)^T + \mathbf{R}_i^+\right) - \mathbf{P}_i(\mathbf{H}_i^+)^T\right)\mathbf{K}^T.\end{aligned}$$

The second term vanished, which gives

$$\mathbf{P}_i^+ = \left(\mathbf{I}_n - \mathbf{K}\mathbf{H}_i^+\right)\mathbf{P}_i, \tag{4.15}$$

where the matrix $\mathbf{K}$ is given by (4.13).

4.1.3 Propagation of estimators and covariance matrices

The last result needed before presenting the Kalman filter concerns the propagation of the estimator $\mathbf{x}_i^+$ and its error covariance matrix $\mathbf{P}_i^+$ from the time t_i to the next time t_{i+1}. The propagation of the estimator $\mathbf{x}_i^+$ to time t_{i+1} is denoted $\mathbf{x}_{i+1}$ and is simply obtained by a model integration from $\mathbf{x}_i^+$

$$\mathbf{x}_{i+1} = \mathbf{M}_{i+1,i}\mathbf{x}_i^+. \tag{4.16}$$

The propagation of the error covariance matrix is denoted $\mathbf{P}_{i+1}$ and is given by

$$\begin{aligned}\mathbf{P}_{i+1} &= E\left[\left(\mathbf{x}_{i+1} - \mathbf{x}_{i+1}^t\right)\left(\mathbf{x}_{i+1} - \mathbf{x}_{i+1}^t\right)^T\right] \\ &= E\left[\mathbf{M}_{i+1,i}\left(\mathbf{x}_i^+ - \mathbf{x}_i^t\right)\left(\mathbf{x}_i^+ - \mathbf{x}_i^t\right)^T\mathbf{M}_{i+1,i}^T\right] \\ &= \mathbf{M}_{i+1,i}E\left[\left(\mathbf{x}_i^+ - \mathbf{x}_i^t\right)\left(\mathbf{x}_i^+ - \mathbf{x}_i^t\right)^T\right]\mathbf{M}_{i+1,i}^T \\ &= \mathbf{M}_{i+1,i}\mathbf{P}_i^+\mathbf{M}_{i+1,i}^T\end{aligned} \tag{4.17}$$

using simple manipulations and (2.1).

4.2 The Kalman filter

The Kalman filter equations can be stated from the results obtained in the previous section. It is a recursive process which assimilates the observations $\mathbf{y}_i$, $i = 1, \ldots, N$, at their correct observation times t_i, $i = 1, \ldots, N$. It involves mainly two steps based on the propagation of an estimator with its error covariance matrix and on the computation of a recursive estimator using the new available observation. We describe these two steps for the first iteration of the Kalman filter which allows to assimilate the first observation vector $\mathbf{y}_1$. At the

initial time t_0, the best estimator is the background vector $\mathbf{x}^b$ with its error covariance matrix $\mathbf{B}$. The *forecast step* propagates to time t_1 both information using (4.16) and (4.17) with $i = 0$, $\mathbf{x}_i^+ = \mathbf{x}^b$ and $\mathbf{P}_i^+ = \mathbf{B}$. The new estimator $\mathbf{x}_1$ is designed as $\mathbf{x}_1^f$ in the Kalman framework to denote the fact that it has been obtained by a forecast. Its error covariance matrix $\mathbf{P}_1$ is denoted $\mathbf{P}_1^f$ for the same reason. The forecast $\mathbf{x}_1^f$ is an estimator of $\mathbf{x}_1^t$ and can be corrected with the observation vector $\mathbf{y}_1$ performing an *analysis step*. Using the formulas (4.13) and (4.14) with $i = 1$, $\mathbf{y}_i^+ = \mathbf{y}_1$, $\mathbf{H}_i^+ = \mathbf{H}_1$, $\mathbf{R}_i^+ = \mathbf{R}_1$, $\mathbf{x}_i = \mathbf{x}_1^f$ and $\mathbf{P}_i = \mathbf{P}_1^f$, we obtain

$$\mathbf{K}_1 = \mathbf{P}_1^f \mathbf{H}_1^T (\mathbf{H}_1 \mathbf{P}_1^f \mathbf{H}_1^T + \mathbf{R}_1)^{-1}$$

and

$$\mathbf{x}_1^+ = \mathbf{x}_1^f + \mathbf{K}_1 (\mathbf{y}_1 - \mathbf{H}_1 \mathbf{x}_1^f).$$

The new estimator $\mathbf{x}_1^+$ is designed as $\mathbf{x}_1^a$ to denote the fact that this estimator has been computed by an analysis step. The corresponding analysis error covariance matrix, denoted $\mathbf{P}_1^a$, is given by (4.15) with $i = 1$, $\mathbf{H}_i^+ = \mathbf{H}_1$ and $\mathbf{P}_i = \mathbf{P}_1^f$. This two-step process can continue recursively by performing a new forecast step to time t_2 from the current analysis $\mathbf{x}_1^a$. The Kalman filter is summarized in Algorithm 4.1 and is illustrated in Figure 4.1.

Algorithm 4.1 Kalman filter with perfect model operators

1: $\mathbf{x}_0^a = \mathbf{x}^b$
2: $\mathbf{P}_0^a = \mathbf{B}$
3: **for** $i = 1$ to N **do**
4: (* **Forecast step** *)
5: $\mathbf{x}_i^f = \mathbf{M}_{i,i-1} \mathbf{x}_{i-1}^a$
6: $\mathbf{P}_i^f = \mathbf{M}_{i,i-1} \mathbf{P}_{i-1}^a \mathbf{M}_{i,i-1}^T$
7: (* **Analysis step** *)
8: $\mathbf{K}_i = \mathbf{P}_i^f \mathbf{H}_i^T (\mathbf{H}_i \mathbf{P}_i^f \mathbf{H}_i^T + \mathbf{R}_i)^{-1}$
9: $\mathbf{x}_i^a = \mathbf{x}_i^f + \mathbf{K}_i (\mathbf{y}_i - \mathbf{H}_i \mathbf{x}_i^f)$
10: $\mathbf{P}_i^a = (\mathbf{I}_n - \mathbf{K}_i \mathbf{H}_i) \mathbf{P}_i^f$
11: **end for**

We can make some comments on the formulation of this algorithm. Since the background $\mathbf{x}^b$ is the best estimator of the state of the system at the initial time, it is denoted as the result of an analysis step, i.e., $\mathbf{x}_0^a$. Therefore, the background error covariance matrix is denoted $\mathbf{P}_0^a$. The computation of the analysis $\mathbf{x}_i^a$ at line 9 is performed in two stages by first computing at line 8 the matrix $\mathbf{K}_i$ called the *Kalman gain* or the *Kalman matrix*.

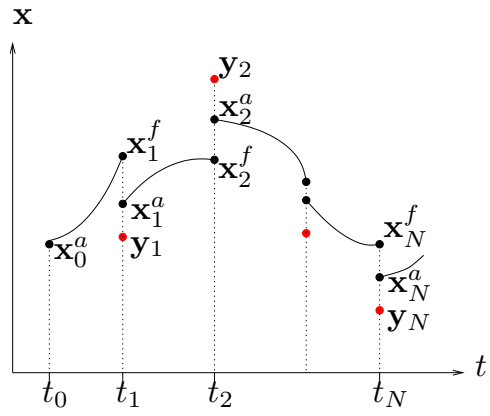

Figure 4.1 — *Kalman filter scheme.*

The Kalman filter has a simple expression and there are a huge amount of engineer applications which use it. An historical use relates to the Apollo mission where a Kalman filter was implemented in 14-bit arithmetic to guide the descent of the lunar module (Cipra, 1993). Nowadays, the Kalman filter is still used mainly for navigational and guidance systems on vehicles such as ships, aircrafts, submarines, guided missiles, and spacecrafts. In these cases, the model operator is computed using an inertial navigation system which computes the position, orientation, and velocity without the need of external references but using accelerometers, gyroscopes and a dead reckoning process. At the same time, observations of positions are coming from a GPS system. The goal of the Kalman filter is to use both informations with their different error characteristics to produce an optimal position estimation (Borre and Strang, 1997, Grewal et al., 2000). As said previously, its applications for data assimilation problems with operational ocean and atmospheric models encounter several major difficulties such as the computational cost, the storage of matrices and the nonlinearities in the operators. We address the first two issues in what follows while the problem of nonlinearities is treated in Chapter 7.

4.2.1 Computational cost

The computational cost of the Kalman filter can be studied by performing an evaluation of the required number of floating-point operations (flop). This study will allow to highlight the most expensive parts of the algorithm. To simplify the count, we assume that each basic operation (addition, multiplication, subtraction and division) takes the same time and is equal to one flop. In general, this assumption is not true and depends on the processor properties and on the precision used for the computation (Overton, 2001).

Before presenting the study, we recall some useful results and comments. Counting the number of flops for a matrix addition or a matrix multiplication is rather simple. The number of flops needed to add two matrices $\mathbf{A}_1 \in \mathbb{R}^{n_1 \times n_2}$ and $\mathbf{A}_2 \in \mathbb{R}^{n_1 \times n_2}$ is equal to $n_1 n_2$ and the number of flops to multiply two matrices $\mathbf{B}_1 \in \mathbb{R}^{n_1 \times n_2}$ and $\mathbf{B}_2 \in \mathbb{R}^{n_2 \times n_3}$ is equal to $2n_1 n_2 n_3$ ($n_1 n_2 n_3$ multiplications and $n_1 n_2 n_3$ additions) (Golub and Van Loan, 1996). However, there is a big gap between the mathematical formulation of an algorithm and an efficient implementation of it. Some cautions must be taken before beginning the implementation since there are tricks that can be used to decrease the number of flops. We review some classical ones. The first trick is based on the fact that the associativity of matrix multiplications does not imply an invariance in the number of flops. Assuming three matrices $\mathbf{B}_1 \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{B}_2 \in \mathbb{R}^{n_2 \times n_3}$ and $\mathbf{B}_3 \in \mathbb{R}^{n_3 \times n_4}$, the associativity property gives

$$(\mathbf{B}_1\mathbf{B}_2)\mathbf{B}_3 = \mathbf{B}_1(\mathbf{B}_2\mathbf{B}_3).$$

The numerical result is independent of the order in which the two matrix products are performed. Nevertheless, the number of flops required is different. For the first expression, the number of flops is given by

$$(\mathbf{B}_1\mathbf{B}_2)\mathbf{B}_3 = 2n_1 n_2 n_3 + 2n_1 n_3 n_4 \tag{4.18}$$

while for the second one

$$\mathbf{B}_1(\mathbf{B}_2\mathbf{B}_3) = 2n_2 n_3 n_4 + 2n_1 n_2 n_4. \tag{4.19}$$

Depending on the size of the matrices, one implementation of this matrix product will be more efficient than the other one.

Another trick is about the matrix inversion. When the inverse of a known matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is involved in an algorithm, the naive approach is to explicitly compute the inverse $\mathbf{A}^{-1}$ and to store it. For a general matrix with no specific properties, it is done by first applying a LU factorization,

$$\mathbf{A} = \mathbf{L}\mathbf{U},$$

and then by inversing the two triangular matrices

$$\mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}.$$

The number of flops for the LU factorization is equal to $\frac{2}{3}n^3$ (Golub and Van Loan, 1996, p.98) and the number of flops for each triangular matrix inversion is given by $\frac{1}{3}n^3 + \frac{3}{4}n^2$ (Quarteroni et al., 2000, p.70). Nevertheless, in most cases, the explicit inversion must be skipped since the algorithm does not need to compute explicitly the inverse of the matrix but needs to compute the inverse of a matrix times a vector, namely $\mathbf{A}^{-1}\mathbf{b}$. This computation is equivalent to solve the linear system

$$\mathbf{A}\mathbf{z} = \mathbf{b}$$

for $\mathbf{z}$. This time, the computation of $\mathbf{z}$ is done using the LU factorization of $\mathbf{A}$ followed by one forward and one backward substitution, for a cost of $2n^2$ flops (Golub and Van Loan, 1996, p.89). We see that it is cheaper to solve a linear system ($\frac{2}{3}n^3 + 2n^2$ flops) than to inverse a matrix and perform the matrix-vector product ($\frac{2}{3}n^3 + \frac{1}{3}n^3 + \frac{3}{4}n^2 + 2n^2$ flops). Moreover, solving the linear system is numerically more accurate. A last trick is to use the properties of the matrices involved in the algorithm such as sparsity, band property, symmetry or positive definiteness. Most of linear algebra operations can be optimized to reduced the number of flops involved in these cases.

Finally, it is important to note that scientific linear algebra packages exist, such as LAPACK (Anderson et al., 1999). This later implements a large number of algebra operations such as solving linear equations, least-squares problems, eigenvalue problems, and singular value problems. It also implements the most famous matrix factorizations (LU, Cholesky, QR, SVD, Schur, etc.). It is preferable to use this kind of packages instead of coding its own routines since they reorganize the underlying algorithms to use block matrix operations. These block operations allow to perform minimal data motions using the multi-layered memory hierarchies of the machines (disk, main memory, cache, register). Moreover, the LAPACK routines are redesigned in the ScaLAPACK package to exploit the power of parallel machines with distributed memory (Blackford et al., 1997).

With these remarks, we are now able to present an efficient way to implement the Kalman filter and to deduce the number of flops for one iteration. For some lines of the algorithm, special attention must be payed and the tricks presented above must be used to reduce the number of flops as far as possible. At lines 8 and 9 of Algorithm 4.1, the Kalman gain matrix $\mathbf{K}_i$ and the analysis $\mathbf{x}_i^a$ are computed. These operations can be set together to obtain

$$\mathbf{x}_i^a = \mathbf{x}_i^f + \mathbf{P}_i^f \times \mathbf{H}_i^T \times (\mathbf{H}_i \mathbf{P}_i^f \mathbf{H}_i^T + \mathbf{R}_i)^{-1} \times (\mathbf{y}_i - \mathbf{H}_i \mathbf{x}_i^f),$$

where the three main products are represented by the symbol *times*. To minimize the computational cost, the products must be performed from right to left since the rightmost factor is a vector of size n. The first rightmost product implies an inverse matrix which must not be explicitly computed. Instead, the following linear system is constructed,

$$\mathbf{A}\mathbf{z} = \mathbf{b},$$

where $\mathbf{A} = \mathbf{H}_i \mathbf{P}_i^f \mathbf{H}_i^T + \mathbf{R}_i$ and $\mathbf{b} = \mathbf{y}_i - \mathbf{H}_i \mathbf{x}_i^f$, and is solved for $\mathbf{z}$. Since $\mathbf{A} \in \mathbb{R}^{p_i \times p_i}$ is symmetric positive definite, a Cholesky factorization requiring only $\frac{1}{3}p_i^3$ flops (Golub and Van Loan, 1996, p.98) can be performed instead of the LU decomposition. Next, at line 10, we have to compute the covariance error matrix

$$\mathbf{P}_i^a = (\mathbf{I}_n - \mathbf{P}_i^f \mathbf{H}_i^T (\mathbf{H}_i \mathbf{P}_i^f \mathbf{H}_i^T + \mathbf{R}_i)^{-1} \mathbf{H}_i) \mathbf{P}_i^f, \tag{4.20}$$

which involves the inverse matrix product $(\mathbf{H}_i\mathbf{P}_i^f\mathbf{H}_i^T + \mathbf{R}_i)^{-1}\mathbf{H}_i$. This product can be computed by solving the following linear system with n right-hand sides,

$$\mathbf{A}\mathbf{Z} = \mathbf{H}_i,$$

for $\mathbf{Z} \in \mathbb{R}^{p_i \times n}$. Since the Cholesky factorization of the matrix $\mathbf{A}$ has already been computed, it will cost $2np_i^2$ flops to compute the n columns of $\mathbf{Z}$ corresponding to n right-hand sides. The computation of $\mathbf{P}_i^f\mathbf{H}_i^T\mathbf{Z}$ involved in (4.20) then requires $2n^2p_i + 2n^2p_i$ flops using (4.18). Thanks to the special structure of an identity matrix, the subtraction in (4.20) costs n flops and the final post-multiplication by $\mathbf{P}_i^f$ costs $2n^3$ flops. The number of flops for each lines of the algorithm 4.1 is summarized in Table 4.1. Another study of the number

Line	Operation	Flops
5	$\mathbf{M}_{i,i-1}\mathbf{x}_{i-1}^a$	$2n^2$
6	$\mathbf{M}_{i,i-1}\mathbf{P}_{i-1}^a\mathbf{M}_{i,i-1}^T$	$4n^3$
8	$\mathbf{A} = \mathbf{H}_i\mathbf{P}_i^f\mathbf{H}_i^T + \mathbf{R}_i$	$2n^2p_i + 2np_i^2 + p_i^2$
9	$\mathbf{b} = \mathbf{y}_i - \mathbf{H}_i\mathbf{x}_i^f$	$2np_i + p_i$
9	$\mathbf{z} = \mathbf{A}^{-1}\mathbf{b}$	$\frac{1}{3}p_i^3 + 2p_i^2$
9	$\mathbf{x}_i^f + \mathbf{P}_i^f\mathbf{H}_i^T\mathbf{z}$	$2np_i + 2n^2 + n$
10	$\mathbf{Z} = \mathbf{A}^{-1}\mathbf{H}_i$	$2np_i^2$
10	$(\mathbf{I}_n - \mathbf{P}_i^f\mathbf{H}_i^T\mathbf{Z})\mathbf{P}_i^f$	$2n^2p_i + 2n^2p_i + n + 2n^3$

Table 4.1 — *Number of flops for one iteration of the Kalman filter.*

of required flops is presented in Lewis and Lakshmivarahan (2008) which gives roughly the same order of flops as ours. However, it is impossible to draw the exact number of flops without defining precisely the methods used to solve the linear systems and the architecture of the machine. When the Kalman filter is implemented for a specific application, the sparsity of some matrices, such as $\mathbf{R}_i$ and $\mathbf{H}_i$, can be exploited to decrease the number of flops. Flop counting is a quick method that capture only one of the several dimensions of the efficiency of an algorithm (Golub and Van Loan, 1996).

In our problems, the size of the state vector, n, is much larger than the size of observation vector, p_i (typically, around three orders of difference). Thus, the most expensive operations are the computation of the covariance matrices $\mathbf{P}_i^f$ and $\mathbf{P}_i^a$ at lines 6 and 10, respectively, since they imply operations with n^3 flops. The other operations involved in the Kalman filter do not overtake n^2p_i or $\frac{1}{3}p_i^3$. The size of the state vector used to represents the state of the atmosphere or of the ocean in operational forecast systems increases constantly over the years. At the ECMWF, the last version of their global atmospheric forecast system (IFS with the T1279L91 configuration) has more than 1 billion of variables with 10 millions of assimilated observations per day (Bonavita and L. Isaksen, 2012). More generally, state vectors with 10 millions of components and observation vectors with 100.000 measurements are common. In

this context, the computation of the forecast error covariance matrix $\mathbf{P}_i^f$ which requires $4n^3$ flops will take more than 4 days on the fastest super computer in the world (assuming that its theoretical peak of 11,28 Peta flops per second can be achieved, see `http://www.top500.org`). This computational cost is not affordable and shows the impractical use of the Kalman filter for operational weather forecast.

4.2.2 Storage

Besides the computational issue, there is also a storage issue with dense matrices of size n. During the past decades the memory of computers has drastically increased. Memory is cheap but still finite. Double-precision floating-point is a format commonly used for scientific computations. It is a binary format that occupies 64 bits (8 bytes) in memory. There is one bit for the sign, 11 bits for the exponent and 52 bits for the significand (IEEE 754). As for computational costs, there are some tricks to save memory. The first one is to eliminate the data redundancy. As an example, the error covariance matrices are symmetric, thus only the upper or the lower triangular part must be stored. The second trick is to replace matrices by algorithms. There are matrices, such as $\mathbf{M}_{i+1,i}$ and $\mathbf{H}_i$, which are only used to perform matrix-vector products. It could be unnecessary to store these matrices explicitly, but rather to implement routines which compute the matrix-vector products. Nevertheless, the Kalman filter does not allow to skip the storage of the error covariance matrices. For problems with 10 millions of variable, the storage of one matrix requires 400 terabytes of memory, which is prohibitive for current computers.

4.2.3 Numerical stability

One last problem of the original Kalman filter is its numerical instability since it may suffer of a lack of robustness against round-off errors. This problem is due to the finite precision used in the implementation and to the mathematical operations that are performed. In the 1960's, when the Kalman filter was introduce in the aerospace industry and in the NASA's space program, the arithmetic precision in the computers and in the embedded systems were lower than now and many numerical problems appeared. Numerical stability can be improved by increasing the arithmetic precision (by adding more bits in the binary format of real numbers), but even with current computers, numerical problems still appear when the Kalman filter is implemented. Two main problems can occur. The first one is that the error covariance matrix, which is symmetric positive definite by definition, could become non symmetric and indefinite. The second one is that even if $\mathbf{H}_i$ has full rank and $\mathbf{P}_i^f$ and $\mathbf{R}_i$ are symmetric positive definite, the matrix $\mathbf{H}_i\mathbf{P}_i^f\mathbf{H}_i^T + \mathbf{R}_i$ could be not invertible

due to round-off errors in the computation of $\mathbf{H}_i\mathbf{P}_i^f\mathbf{H}_i^T$.

Square root filtering is a way to increase the precision of the Kalman filter when no more hardware precision is available. The first *square root Kalman filter* seems to have been developed by J. Potter (Battin, 1964) and extended to handle model noise (Andrews, 1968) and observation vectors instead of scalar measurement (Bellantoni and Dodge, 1968). The fundamental concept of square root filtering is to factorize the initial error covariance matrix $\mathbf{B}$. Since it is a symmetric positive definite matrix, a Cholesky factorization can be applied,

$$\mathbf{B} = \mathbf{C}\mathbf{C}^T,$$

where $\mathbf{C} \in \mathbb{R}^{n\times n}$ is a lower triangular matrix called the square root. The matrix $\mathbf{C}$ is then used to propagate the error statistics instead of the full error covariance matrix. The first advantage of this reformulation is that the product $\mathbf{C}\mathbf{C}^T$ is always symmetric non-negative definite. The second advantage is about the condition number of the matrices involved in the algorithm. Since the condition number of $\mathbf{C}$ is the square root of the condition number of $\mathbf{B}$, it may improve the precision of the algorithm. A famous implementation of this approach is the Carlson-Schmidt square root filter. It computes the analysis step using a fast triangular update (Carlson, 1973) and the forecast step using Householder reflections (Golub and Van Loan, 1996).

There is an alternative to the Cholesky factorization which leads up to another square root Kalman filter. It is based on a modified Cholesky factorization given by

$$\mathbf{B} = \mathbf{F}\mathbf{D}\mathbf{F}^T, \tag{4.21}$$

where $\mathbf{F}$ is a unit upper triangular matrix (its diagonal elements are equal to 1) and $\mathbf{D}$ is a diagonal matrix. There is no huge difference with the Cholesky factorization except that this algorithm does not require to compute square roots of numbers. A common implementation of the square root Kalman filter, using (4.21) as initial decomposition, is the Bierman-Thornton filter (Bierman, 1977, Thornton, 1976).

The computational cost of the square root filters does not exceed the cost of the conventional Kalman filter by more than 50 percent in most practical problems. Nevertheless, the square root filters can yield twice the effective precision of the Kalman filter for ill-conditioned problems (Kaminski, 1971).

4.3 The SEEK filter

In this section, the *Singular Evolutive Extended Kalman filter* (SEEK filter) proposed by Pham et al. (1998) is presented. It uses a framework similar to that of the square root Kalman filter ensuring a sufficient numerical stability. Moreover, it solves the computational and storage issues of the error covariance

matrices constraining these ones to be of low rank. This constraint is well motivated. Indeed, the initial error covariance matrix $\mathbf{B}$ is a huge matrix which contains the error statistics for each component of the background vector. Its computation is not usually trivial and suitable processes are used to model the background error covariance matrix (see Section 3.5). The error covariance matrix $\mathbf{B}$ contains uncertainties in the error statistics and thus it will be inappropriate to spend a huge amount of computational time to propagate probably erroneous informations. For this reason, using low rank approximations seems to be appropriate.

4.3.1 Derivation of the algorithm

The background error covariance matrix $\mathbf{B}$ can be diagonalized by a spectral decomposition,

$$\mathbf{B} = \mathbf{L}_0\mathbf{U}_0\mathbf{L}_0^T + \hat{\mathbf{L}}_0\hat{\mathbf{U}}_0\hat{\mathbf{L}}_0^T, \tag{4.22}$$

where the diagonal matrix $\mathbf{U}_0 \in \mathbb{R}^{r\times r}$ contains the r largest eigenvalues of $\mathbf{B}$ and the diagonal matrix $\hat{\mathbf{U}}_0 \in \mathbb{R}^{(n-r)\times(n-r)}$ contains the remaining eigenvalues, while the matrices $\mathbf{L}_0 \in \mathbb{R}^{n\times r}$ and $\hat{\mathbf{L}}_0 \in \mathbb{R}^{n\times(n-r)}$ are formed with the corresponding eigenvectors. Since the matrix $\mathbf{B}$ is symmetric, these eigenvectors can be chosen such that they are orthogonal to each other and have norm one, i.e., $[\mathbf{L}_0\hat{\mathbf{L}}_0]^T[\mathbf{L}_0\hat{\mathbf{L}}_0] = \mathbf{I}_n$. The idea of the SEEK filter is to keep only the r first eigenvectors and to discard the $n-r$ last eigenvectors for the initial analysis covariance matrix, i.e.,

$$\mathbf{P}_0^a = \mathbf{L}_0\mathbf{U}_0\mathbf{L}_0^T. \tag{4.23}$$

We obtain a decomposition which is similar to the modified Cholesky factorization (4.21) except that it is a low rank approximation of $\mathbf{B}$ where $\mathbf{L}_0 \in \mathbb{R}^{n\times r}$ is a rectangular matrix with r columns.

In the following, we derive the equations of the SEEK filter for the first iteration using the equation of the Kalman filter and the low rank decomposition (4.23). The initial analysis is kept unchanged and is equal to the background vector $\mathbf{x}^b$. It evolves like as in the Kalman filter. The evolution of the initial error covariance matrix becomes

$$\begin{aligned}\mathbf{P}_1^f =& \mathbf{M}_{1,0}\mathbf{P}_0^a\mathbf{M}_{1,0}^T\\ =& \mathbf{M}_{1,0}\mathbf{L}_0\mathbf{U}_0\mathbf{L}_0^T\mathbf{M}_{1,0}^T\\ =& \mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T,\end{aligned} \tag{4.24}$$

where $\mathbf{L}_1 = \mathbf{M}_{1,0}\mathbf{L}_0$ is formed by integrating the columns of $\mathbf{L}_0$ to time t_1. The Kalman gain can be rewritten from its definition (line 8 of Algorithm 4.1) using the equality (4.24). It gives the following expression

$$\mathbf{K}_1 = \mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T(\mathbf{H}_1\mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T + \mathbf{R}_1)^{-1},$$

which can be reformulated as

$$\mathbf{K}_1 = \mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T\left[\mathbf{R}_1^{-1} - \mathbf{R}_1^{-1}\mathbf{H}_1\mathbf{L}_1(\mathbf{U}_0^{-1} + \mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1}\mathbf{H}_1\mathbf{L}_1)^{-1}\mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1}\right]$$

using the Sherman–Morrison–Woodbury formula (A.1). The following equality

$$(\mathbf{R}_1 + \mathbf{H}_1\mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T)^{-1}\mathbf{H}_1\mathbf{L}_1\mathbf{U}_0 = \mathbf{R}_1^{-1}\mathbf{H}_1\mathbf{L}_1(\mathbf{U}_0^{-1} + \mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1}\mathbf{H}_1\mathbf{L}_1)^{-1},$$

obtained from (A.2), allows to write the Kalman gain as

$$\mathbf{K}_1 = \mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T\left[\mathbf{R}_1^{-1} - (\mathbf{R}_1 + \mathbf{H}_1\mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T)^{-1}\mathbf{H}_1\mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1}\right]$$

and to factorize it as

$$\mathbf{K}_1 = \mathbf{L}_1\left[\mathbf{U}_0 - \mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T(\mathbf{R}_1 + \mathbf{H}_1\mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T)^{-1}\mathbf{H}_1\mathbf{L}_1\mathbf{U}_0\right]\mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1}.$$

Applying, in a reverse way, the Sherman–Morrison–Woodbury formula to the above equation, we have

$$\mathbf{K}_1 = \mathbf{L}_1\left[(\mathbf{U}_0^{-1} + \mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1}\mathbf{H}_1\mathbf{L}_1)^{-1}\right]\mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1}.$$

The final expression of the Kalman gain is given by

$$\mathbf{K}_1 = \mathbf{L}_1\mathbf{U}_1\mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1}, \tag{4.25}$$

where

$$\mathbf{U}_1 = (\mathbf{U}_0^{-1} + \mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1}\mathbf{H}_1\mathbf{L}_1)^{-1}. \tag{4.26}$$

Lastly, the analysis error covariance matrix P_1^a, defined at line 10 of Algorithm 4.1, can be rewritten as

$$\begin{aligned}\mathbf{P}_1^a =& \mathbf{P}_1^f - \mathbf{K}_1\mathbf{H}_1\mathbf{P}_1^f \\ =& \mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T - \mathbf{L}_1(\mathbf{U}_0^{-1} + \mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1}\mathbf{H}_1\mathbf{L}_1)^{-1}\mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1}\mathbf{H}_1\mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T,\end{aligned}$$

using (4.24) and (4.25). From the formula (A.2), the following equality can be deduced,

$$\begin{aligned}(\mathbf{U}_0^{-1} + \mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1}\mathbf{H}_1\mathbf{L}_1)^{-1}\mathbf{L}_1^T\mathbf{H}_1^T\mathbf{R}_1^{-1} \\ = \mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T(\mathbf{R}_1 + \mathbf{H}_1\mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T)^{-1}\end{aligned}$$

which allows to write the analysis error covariance matrix as

$$\mathbf{P}_1^a = \mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T - \mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T(\mathbf{R}_1 + \mathbf{H}_1\mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T)^{-1}\mathbf{H}_1\mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T.$$

Factorizing this expression as

$$\mathbf{P}_1^a = \mathbf{L}_1\left[\mathbf{U}_0 - \mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T(\mathbf{R}_1 + \mathbf{H}_1\mathbf{L}_1\mathbf{U}_0\mathbf{L}_1^T\mathbf{H}_1^T)^{-1}\mathbf{H}_1\mathbf{L}_1\mathbf{U}_0\right]\mathbf{L}_1^T,$$

Algorithm 4.2 SEEK filter with perfect model operators

1: $\mathbf{x}_0^a = \mathbf{x}^b$
2: $\mathbf{P}_0^a = \mathbf{L}_0\mathbf{U}_0\mathbf{L}_0^T$
3: **for** $i = 1$ to N **do**
4: (* **Forecast step** *)
5: $\mathbf{x}_i^f = \mathbf{M}_{i,i-1}\mathbf{x}_{i-1}^a$
6: $\mathbf{L}_i = \mathbf{M}_{i,i-1}\mathbf{L}_{i-1}$
7: $\mathbf{P}_i^f = \mathbf{L}_i\mathbf{U}_{i-1}\mathbf{L}_i^T$
8: (* **Analysis step** *)
9: $\mathbf{U}_i^{-1} = \mathbf{U}_{i-1}^{-1} + \mathbf{L}_i^T\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i\mathbf{L}_i$
10: $\mathbf{K}_i = \mathbf{L}_i\mathbf{U}_i\mathbf{L}_i^T\mathbf{H}_i^T\mathbf{R}_i^{-1}$
11: $\mathbf{x}_i^a = \mathbf{x}_i^f + \mathbf{K}_i(\mathbf{y}_i - \mathbf{H}_i\mathbf{x}_i^f)$
12: $\mathbf{P}_i^a = \mathbf{L}_i\mathbf{U}_i\mathbf{L}_i^T$
13: **end for**

and using the Sherman–Morrison–Woodbury formula (A.1), we obtain

$$\mathbf{P}_1^a = \mathbf{L}_1\mathbf{U}_1\mathbf{L}_1, \tag{4.27}$$

from (4.26) The expression of the analysis error covariance matrix (4.27) after one iteration has the same form has the initial one, meaning that the process can be conducted recursively. The equations are summarized in Algorithm 4.2.

The use of low rank error covariance matrices has the consequence of correcting the state vector only in the columns of $\mathbf{L}_i$ (see line 11 of Algorithm 4.2 with the definition of the Kalman gain at line 10). There are physical considerations in oceanography which support this approximation. Indeed, the ocean is basically a dissipative dynamical system that exhibits an attractor, meaning that asymptotically the trajectories of the state vector belong only to a small part of the phase space. The existence of a global attractor has been proved for the Navier-Stokes equations (Lions et al., 1997). In the vicinity of this attractor, orthogonal perturbations will be naturally damped while tangent perturbations will not. The efficiency of the SEEK filter comes from the fact that the matrix $\mathbf{L}_i$ contains the main directions of variability tangent to the attractor.

4.3.2 Computational cost

As for the Kalman filter, the computational cost of the SEEK filter can be studied by counting the required number of flops for one iteration. Before presenting the result, some comments must be done on certain parts of Algorithm 4.2. These comments are based on the same counting rules and tricks

as the ones presented in Subsection 4.2.1. The main point is about the small square matrix $\mathbf{U}_i$. Since this matrix and its inverse both appear at different lines of the algorithm, we have to choose if we store explicitly $\mathbf{U}_i$ or $\mathbf{U}_i^{-1}$. The obvious choice is to store $\mathbf{U}_i^{-1}$ for each iteration because the updating rule at line 9 is given for the matrix inverse. With this choice, we have to inverse the initial matrix $\mathbf{U}_0$. Fortunately, it can be performed using only r flops since it is a diagonal matrix. At line 7, the product $\mathbf{L}_i\mathbf{U}_{i-1}\mathbf{L}_i^T$ is computed in two stages. Firstly, the following linear system with n right-hand sides,

$$\mathbf{U}_{i-1}^{-1}\mathbf{Z} = \mathbf{L}_i^T,$$

is solved for $\mathbf{Z} \in \mathbb{R}^{r\times n}$. It requires $\frac{1}{3}r^3$ flops to factorize $\mathbf{U}_{i-1}^{-1}$ and $2nr^2$ to compute the n columns of $\mathbf{Z}$. Secondly, the product $\mathbf{L}_i\mathbf{Z}$ is performed using $2n^2r$ flops. Note that the computation of the error covariance matrices, namely $\mathbf{P}_i^f$ and $\mathbf{P}_i^a$, are not mandatory for the computation of the rest of the algorithm. They can thus be skipped if no user-oriented diagnostic are required. The update of $\mathbf{U}_i^{-1}$ can be rewritten as

$$\mathbf{U}_i^{-1} = \mathbf{U}_{i-1}^{-1} + (\mathbf{H}_i\mathbf{L}_i)^T\mathbf{R}_i^{-1}(\mathbf{H}_i\mathbf{L}_i).$$

To reduce the number of flops, the operations must be calculated using the following order. The matrix computation $\mathbf{H}_i\mathbf{L}_i$ which required $2np_ir$ flops is first performed. Then the linear system with r multiple right hand sides

$$\mathbf{R}_i\mathbf{Z} = \mathbf{H}_i\mathbf{L}_i$$

is solved for $\mathbf{Z} \in \mathbb{R}^{p\times r}$ which required $\frac{1}{3}p_i^3 + 2p_i^2r$ flops. The solution is used to compute the product $(\mathbf{H}_i\mathbf{L}_i)^T\mathbf{Z}$ and the matrix addition is performed. These two operations necessitate $2p_ir^2$ and r^2 flops, respectively.

The last comment is about the computation of the analysis vector. Combining the lines 10 and 11 of the algorithm, we have

$$\mathbf{x}_i^a = \mathbf{x}_i^f + \mathbf{L}_i\mathbf{U}_i(\mathbf{H}_i\mathbf{L}_i)^T\mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{H}_i\mathbf{x}_i^f).$$

To minimize the number of flops, the matrix multiplications must be performed from right to left once the departure vector $\mathbf{y}_i - \mathbf{H}_i\mathbf{x}_i^f$ is computed. The multiplications by $\mathbf{R}_i^{-1}$ and $\mathbf{U}_i$ are performed by solving a linear system. Note that the Cholesky factorization of $\mathbf{R}_i^{-1}$ is already computed. The flops count for each lines of the SEEK algorithm is summarized in Table 4.2.

In our data assimilation problems, the size of the state vector, n, is much larger than the size of the observation vector, p_i, which is larger than the size, r, of the subspace $\mathbf{L}_0$. From Table 4.2, the most expensive operation in the SEEK filter is the update of $\mathbf{L}_{i-1}$ at line 6 since it implies $2n^2r$ flops. A usual size for the subspace in operational oceanographic forecast is around $r = 50$. With a 10 million component state vector, this operation will take less than one second on the fastest computer in the world. It demonstrates that the SEEK

Line	Operation	Flops
5	$\mathbf{M}_{i,i-1}\mathbf{x}^a_{i-1}$	$2n^2$
6	$\mathbf{M}_{i,i-1}\mathbf{L}_{i-1}$	$2n^2r$
7	$\mathbf{L}_i\mathbf{U}_{i-1}\mathbf{L}_i^T$	$\frac{1}{3}r^3 + 2nr^2 + 2n^2r$
9	$\mathbf{U}_{i-1}^{-1} + \mathbf{L}_i^T\mathbf{H}_i^T\mathbf{R}_i^{-1}\mathbf{H}_i\mathbf{L}_i$	$2np_ir + \frac{1}{3}p_i^3 + 2p_i^2r + 2p_ir^2 + r^2$
11	$\mathbf{b} = \mathbf{y}_i - \mathbf{H}_i\mathbf{x}_i^f$	$2np_i + p_i$
11	$\mathbf{z} = \mathbf{R}_i^{-1}\mathbf{b}$	$2p_i^2$
11	$\mathbf{x}_i^f + \mathbf{L}_i\mathbf{U}_i(\mathbf{H}_i\mathbf{L}_i)^T\mathbf{z}$	$2rp_i + \frac{1}{3}r^3 + 2r^2 + 2nr + n$
12	$\mathbf{L}_i\mathbf{U}_i\mathbf{L}_i^T$	$\frac{1}{3}r^3 + 2nr^2 + 2n^2r$

Table 4.2 — *Number of flops for one iteration of the SEEK filter.*

filter has drastically reduced the computational cost compared to the Kalman filter. Moreover, the storage issue is also solved since the largest matrix that must be stored is $\mathbf{L}_i$ and represents only 4Gb of memory.

The computational cost of the spectral decomposition of $\mathbf{B}$ given by (4.22) is not outlined in this section. One can remark that the filter requires only the computation of the first r eigenvectors related to the r largest eigenvalues. Since $\mathbf{B}$ is assumed to be a large sparse symmetric positive definite matrix, iterative methods such as the Lanczos method or the Jacobi-Davidson method are well suited to achieve this task (Bai et al., 1987). Their convergence rate and the underlying number of flops depend on the matrix in itself. A flop counting is thus irrelevant in this case. However, the next section presents a practical application of the SEEK filter where a flop counting for the spectral decomposition is available.

4.3.3 A practical application of the SEEK filter

The equations of the SEEK filter have been derived in Subsection 4.3.1 for a general background error covariance matrix. However, if the model is sufficiently good and representative of the system variability, the background error covariance matrix can be modeled using informations from a set of state vectors computed by a model integration. This approach is often used in the context of the twin experiment since the model is unbiased and the observations are simulated from a model trajectory with known statistical properties (Brasseur and Verron, 2006). In the following, we present how to apply a SEEK filter in such a situation and we address the problem of choosing the dimension of the subspace.

Empirical orthogonal functions

To build a background error covariance matrix from a model information, one considers a set of m (linearly independent) state vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$,

taken either from previous data assimilation windows or from a free model run. These vectors are supposed to be representative of the variability of the system. Regarding now the state vector $\mathbf{x}$ as a random vector and considering that the set $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ is a sample of realizations, one calculates the sample covariance matrix

$$\mathbf{S} = \frac{1}{m-1} \sum_{i=1}^{m} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T = \mathbf{X}\mathbf{X}^T, \tag{4.28}$$

where $\bar{\mathbf{x}}$ is the sample mean and $\mathbf{X} = [\mathbf{x}_1/\sqrt{m-1}, \ldots, \mathbf{x}_m/\sqrt{m-1}] \in \mathbb{R}^{n \times m}$. The spectral decomposition of $\mathbf{S}$ can be defined as

$$\mathbf{S} = \mathbf{L}_0 \mathbf{U}_0 \mathbf{L}_0^T + \hat{\mathbf{L}}_0 \hat{\mathbf{U}}_0 \hat{\mathbf{L}}_0^T, \tag{4.29}$$

where the diagonal matrix $\mathbf{U}_0 \in \mathbb{R}^{r \times r}$ contains the r largest eigenvalues of $\mathbf{S}$ and the diagonal matrix $\hat{\mathbf{U}}_0 \in \mathbb{R}^{(n-r)\times(n-r)}$ contains the remaining eigenvalues, while the matrices $\mathbf{L}_0 \in \mathbb{R}^{n \times r}$ and $\hat{\mathbf{L}}_0 \in \mathbb{R}^{n \times (n-r)}$ are formed with the corresponding eigenvector. In this case, these eigenvectors are called *Empirical Orthogonal functions* (EOFs) (see, Hannachi et al., 2007, for a review). Computing such a decomposition is called an *EOF analysis*, also known as a *principal component analysis* (PCA)(see Hannachi et al. (2007), for a review). The subspace $\mathbf{L}_0$ is optimal in some sense since it minimizes the reconstruction error (the squared distance between the original trajectory and its orthogonal projection on the subspace),

$$\mathbf{L}_0 = \underset{\mathbf{L} \in \mathbb{R}^{n \times r}}{\operatorname{argmin}} \sum_{i=1}^{m} ||\mathbf{x}_i - \mathbf{L}\mathbf{L}^T \mathbf{x}_i||^2$$

for a given sample of realizations (Jolliffe, 2002). Instead of executing the SEEK filter where the initial analysis error covariance matrix, P_0^a, is based on the spectral decomposition of $\mathbf{B}$ given by (4.22), one can choose to use the spectral decomposition of $\mathbf{S}$ given by (4.29). It is more practical since the structure of $\mathbf{S}$ allows to compute its eigenvectors and eigenvalues cheaply without explicitly storing $\mathbf{S}$. Indeed, The singular value decomposition (SVD) of $\mathbf{X} \in \mathbb{R}^{n \times m}$ is given by

$$\mathbf{X} = \mathbf{T}\mathbf{\Sigma}\mathbf{V}^T, \tag{4.30}$$

where the columns of the orthogonal matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ are the left singular vectors, the columns of the orthogonal matrix $\mathbf{V} \in \mathbb{R}^{m \times m}$ are the right singular vectors and $\mathbf{\Sigma} \in \mathbb{R}^{n \times m}$ is a diagonal matrix containing the singular values of $\mathbf{X}$. A useful linear algebra result claims that the left singular vectors of $\mathbf{X}$ coincide with the eigenvector of $\mathbf{S}$. Moreover, the singular values of $\mathbf{X}$ correspond to the square roots of the eigenvalues of $\mathbf{S}$ (Golub and Van Loan, 1996, M. Wall, 2003). The matrix $\mathbf{L}_0$ can thus be constructed from left singular vectors of $\mathbf{X}$. Essentially, $6nm^2 + 11m^3$ flops are necessary to compute all singular values

and the related left singular vectors (see Golub and Van Loan, 1996, p.254). This computation is affordable for two reasons. On the one hand, the typical size of m is much smaller than n, e.g., it is equal to 500 in operational ocean models such as OPA. On the other hand, the decomposition is computed only once before the beginning of the data assimilation process.

Since the 1940's, EOFs have been used in atmospheric science to search for low dimensional subspaces of the state space in which the dynamics of interest of the original system are contained (Lall et al., 1999). Variants of EOFs-based methods have been (and are still) developed to overcome some difficulties. For instance, the computational cost of the EOFs can be reduced by using the Online Singular Value Decomposition, an incremental SVD algorithm that avoids storing the matrix $\mathbf{X}$ see (see Brand, 2003). Methods such as Rotated EOFs (REOF) (Horel, 1981) or Least Absolute Shrinkage and Selection Operator (LASSO) (Jolliffe et al., 2003) have been proposed to relax the orthogonality condition, while the Extended EOFs (EEOF) attempt to incorporate temporal correlation to the spatial correlation already present in the EOFs, see Weare and Nasstrom (1982).

The EOFs approach is not the only way to compute reduced subspace for suboptimal Kalman filter. It is possible to use, for instance, singular vectors, Liapunov vectors or bred grown vectors. These are computed using information from the model and contain, in a sense, the main directions of variability of the system. Durbiano (2001) performed a study of these families of vectors. When the data assimilation is performed on a shallow water model, she concluded to the clear superiority of the EOFs basis with regards to the other subspaces.

Dimension of the subspace

The number of selected EOFs in the construction of the subspace $\mathbf{L}_0$ is a last important question of practical interest. Theoretically, we have seen that the filter may only perform corrections in the directions tangent to the attractor since the error in other directions will be damped by the model integration. The number of these direction is equal to the number of eigenvalues of the model operator, $\mathbf{M}_{i+1,i}$, having an absolute value larger than or equal to one (Pham et al., 1998). In operational forecast, it is impossible to compute the eigenvalues of this matrix and thus a more practical guideline must be set up. It is often based on the percentage of variation accounted for by the first r EOFs, defined as

$$100 \frac{\sum_{i=1}^{r} \boldsymbol{\lambda}_i}{\sum_{i=1}^{n} \boldsymbol{\lambda}_i}, \tag{4.31}$$

where the $\boldsymbol{\lambda}_i$'s are the eigenvalues of $\mathbf{S}$ in decreasing order. A threshold of variance (e.g., 80 %) explained by the first r EOFs provides a criterion to determine the value of r. It will be often a good compromise since it allows to account for most of the variation without increasing too much the dimension of the subspace spanned by the selected EOFs.

Chapter 5

Connections between the variational and the sequential approaches

The connections between the sequential and the variational methods have been known for long. For linear model and observation operators, both the 4D-Var formulation and the Kalman filter yield the same estimates of the system state (theoretically) at the end of the assimilation window, as shown in Strang and Borre (1997) and in Li and Navon (2001). Their proofs are based on a block matrix and on a statistical view, respectively. In this section, we propose an alternative proof for the equivalence[1] between the Kalman filter and the 4D-Var in the specific case of a perfect linear model operator and of a linear observation operator. This proof, which is based on quadratic optimization techniques, is both short and elementary. Moreover, we use some results from this proof to establish in Section 5.2 a useful connection between a reduced order 4D-Var problem and the SEEK filter. This last result will motivate the development of some preconditioning techniques presented in Chapter 6.

5.1 Connection between 4D-Var / Kalman filter

The goal of this section is to show the connection between the solution of the 4D-Var problem (3.5) and the Kalman filter presented in Algorithm 4.1 under the assumption of linear operators. For more convenience, the 4D-Var problem

[1] By the word «equivalence» we mean, throughout this work, the fact that the considered methods produce the same results at the end of the assimilation window.

(3.5) with linear operator is reformulate as

$$\min_{\mathbf{x}\in\mathbb{R}^n} J^{4\mathtt{D}}(\mathbf{x}) = \frac{1}{2}||\mathbf{x}-\mathbf{x}^b||^2_{\mathbf{B}^{-1}} + \frac{1}{2}\sum_{i=1}^{N}||\mathbf{y}_i - \mathbf{H}_i\mathbf{M}_{i,0}\mathbf{x}||^2_{\mathbf{R}_i^{-1}}, \tag{5.1}$$

using the weighted norms defined by $||\mathbf{z}||^2_{\mathbf{B}^{-1}} = \mathbf{z}^T\mathbf{B}^{-1}\mathbf{z}$ and $||\mathbf{z}||^2_{\mathbf{R}^{-1}} = \mathbf{z}^T\mathbf{R}^{-1}\mathbf{z}$ with the equality $\mathbf{M}_{i,0} = \prod_{j=0}^{j=i-1}\mathbf{M}_{j+1,j}$.

Theorem 5.1 *Suppose that the model operator is perfect and linear, that the observation operator is linear and that the same background $\mathbf{x}^b$, observations $\mathbf{y}_i$, background covariance matrix $\mathbf{B}$, and observation covariance matrices $\mathbf{R}_i$, are given. Then the analysis state $\mathbf{x}_i^a$ computed at time t_i by the Kalman filter and the solution produced by the 4D-Var method, using the first i observations and integrated up to time t_i, are identical and have same covariance matrices. Both methods hence produce the same results at the end of the assimilation window.*

Proof. — In order to be as concise as possible, we set $\mathbf{M}_i = \mathbf{M}_{i,0}$, $\mathbf{H} = \mathbf{H}_i$ and $\mathbf{R} = \mathbf{R}_i$ throughout this proof. Generalizing it would just result in heavier notation. We first introduce the notation $\mathbf{x}_l^k$ to denote the solution of the 4D-Var (5.1) using the first l observations,

$$\min_{\mathbf{x}\in\mathbb{R}^n} J_l(\mathbf{x}) = \frac{1}{2}||\mathbf{x}-\mathbf{x}^b||^2_{\mathbf{B}^{-1}} + \frac{1}{2}\sum_{i=1}^{l}||\mathbf{y}_i - \mathbf{H}\mathbf{M}_i\mathbf{x}||^2_{\mathbf{R}^{-1}},$$

and integrated up to time t_k, and $\mathbf{P}_l^k$ to denote the covariance matrix of $\mathbf{x}_l^k$. We thus want to prove, for each observation time $l = 1, \ldots, N$, that the solution of the 4D-Var using the first l observations and integrated up to time t_l is equal to the analysis produced by the Kalman filter after l iterations, i.e., $\mathbf{x}_l^l = \mathbf{x}_l^a$, and that both state vectors have the same covariance matrices, i.e., $\mathbf{P}_l^l = \mathbf{P}_l^a$.

The proof is by induction. To prove that $\mathbf{x}_1^1 = \mathbf{x}_1^a$ and $\mathbf{P}_1^1 = \mathbf{P}_1^a$, we first notice that $\mathbf{x}_0^0 = \mathbf{x}^b = \mathbf{x}_0^a$ since $\mathbf{x}^b$ is the solution of the 4D-Var problem using no observations. Therefore, the covariance matrix of $\mathbf{x}_0^0$ is given by $\mathbf{P}_0^0 = \mathbf{B} = \mathbf{P}_0^a$. We next define the 4D-Var problem using only the first observation $\mathbf{y}_1$ at time t_1 as

$$\min_{\mathbf{x}\in\mathbb{R}^n} J_1(\mathbf{x}) = \frac{1}{2}||\mathbf{x}-\mathbf{x}_0^0||^2_{(\mathbf{P}_0^0)^{-1}} + \frac{1}{2}||\mathbf{y}_1 - \mathbf{H}\mathbf{M}_1\mathbf{x}||^2_{\mathbf{R}^{-1}}. \tag{5.2}$$

The solution $\mathbf{x}_1^0$ of this problem is computed by nullifying its gradient,

$$(\mathbf{P}_0^0)^{-1}(\mathbf{x}_1^0 - \mathbf{x}_0^0) - \mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}_1 - \mathbf{H}\mathbf{M}_1\mathbf{x}_1^0) = 0. \tag{5.3}$$

Defining

$$\mathbf{W} \equiv (\mathbf{P}_0^0)^{-1} + \mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{M}_1, \tag{5.4}$$

we obtain, by simple extraction of the vector $\mathbf{x}_1^0$ from (5.3)

$$\mathbf{x}_1^0 = \mathbf{W}^{-1}\left(\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}_1 + (\mathbf{P}_0^0)^{-1}\mathbf{x}_0^0\right). \tag{5.5}$$

Note that the matrix $\mathbf{W}$ is invertible since it is the sum of a positive definite and a positive semidefinite matrix, which gives a positive definite matrix. From the Sherman-Morisson-Woodbury formula (see, for example, Horn and Johnson, 2006, p.18), we have that

$$\mathbf{W}^{-1} = \mathbf{P}_0^0 - \mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}(\mathbf{R} + \mathbf{H}\mathbf{M}_1\mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})^{-1}\mathbf{H}\mathbf{M}_1\mathbf{P}_0^0. \tag{5.6}$$

Using this formula in (5.5), developing the product and reordering the terms, we obtain

$$\begin{aligned}\mathbf{x}_1^0 = \mathbf{x}_0^0 + \mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}_1 - \mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}(\mathbf{R} + \mathbf{H}\mathbf{M}_1\mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})^{-1} \\ \mathbf{H}\mathbf{M}_1\mathbf{P}_0^0\left(\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}_1 + (\mathbf{P}_0^0)^{-1}\mathbf{x}_0^0\right).\end{aligned} \tag{5.7}$$

Observing that

$$\begin{aligned}\mathbf{H}\mathbf{M}_1\mathbf{P}_0^0\left(\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}_1 + (\mathbf{P}_0^0)^{-1}\mathbf{x}_0^0\right) = \\ (\mathbf{H}\mathbf{M}_1\mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}} + \mathbf{R})\mathbf{R}^{-1}\mathbf{y}_1 - \mathbf{y}_1 + \mathbf{H}\mathbf{M}_1\mathbf{x}_0^0,\end{aligned}$$

we obtain, after appropriate simplifications,

$$\mathbf{x}_1^0 = \mathbf{x}_0^0 + \mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}(\mathbf{R} + \mathbf{H}\mathbf{M}_1\mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})^{-1}(\mathbf{y}_1 - \mathbf{H}\mathbf{M}_1\mathbf{x}_0^0) \tag{5.8}$$

as solution of the 4D-Var (5.2). If we next integrate this solution $\mathbf{x}_1^0$ up to the first time-step, we find

$$\mathbf{x}_1^1 = \mathbf{M}_1\mathbf{x}_0^0 + \mathbf{M}_1\mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}(\mathbf{R} + \mathbf{H}\mathbf{M}_1\mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})^{-1}(\mathbf{y}_1 - \mathbf{H}\mathbf{M}_1\mathbf{x}_0^0).$$

It thus follows from Algorithm 4.1 that $\mathbf{x}_1^1 = \mathbf{x}_1^a$, since $\mathbf{M}_1\mathbf{x}_0^0 = \mathbf{M}_1\mathbf{x}_0^a = \mathbf{x}_1^f$, $\mathbf{M}_1\mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}} = \mathbf{M}_1\mathbf{P}_0^a\mathbf{M}_1^{\mathrm{T}} = \mathbf{P}_1^f$ and since we use the same observation $\mathbf{y}_1$. Moreover, observe from (5.3) that $\mathbf{W}$, as defined in (5.4), is the Hessian matrix of J_1. Following Rabier and Courtier (1992), who proved that the covariance matrix of a 4D-Var solution is equal to the Hessian inverse, we can thus conclude that the covariance matrix of $\mathbf{x}_1^0$ is given by

$$\mathbf{P}_1^0 = \mathbf{W}^{-1}. \tag{5.9}$$

We have that the covariance matrix of $\mathbf{x}_1^1$, denoted $\mathrm{Cov}(\mathbf{x}_1^1, \mathbf{x}_1^1)$ and defined by $E[(\mathbf{x}_1^1 - \mathbf{x}_1^t)(\mathbf{x}_1^1 - \mathbf{x}_1^t)^T]$, is given by

$$\begin{aligned}\mathbf{P}_1^1 &= \mathrm{Cov}(\mathbf{x}_1^1, \mathbf{x}_1^1) = \mathrm{Cov}(\mathbf{M}_1\mathbf{x}_1^0, \mathbf{M}_1\mathbf{x}_1^0) \\ &= \mathbf{M}_1\mathrm{Cov}(\mathbf{x}_1^0, \mathbf{x}_1^0)\mathbf{M}_1^{\mathrm{T}} = \mathbf{M}_1\mathbf{P}_1^0\mathbf{M}_1^{\mathrm{T}}.\end{aligned}$$

By (5.6) and (5.9), we deduce from this last expression that

$$\mathbf{P}_1^1 = \mathbf{M}_1\mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}} - \mathbf{M}_1\mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}(\mathbf{R} + \mathbf{H}\mathbf{M}_1\mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})^{-1}\mathbf{H}\mathbf{M}_1\mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}},$$

which corresponds to the analysis error covariance matrix $\mathbf{P}_1^a$ produced by the Kalman filter (see Algorithm 4.1), since $\mathbf{M}_1\mathbf{P}_0^0\mathbf{M}_1^{\mathrm{T}} = \mathbf{P}_1^f$.

Now, we suppose that when the j first observations vectors are available, the solution $\mathbf{x}_j^0$ of the corresponding 4D-Var problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} J_j(\mathbf{x}) = \frac{1}{2}||\mathbf{x} - \mathbf{x}_0^0||^2_{(\mathbf{P}_0^0)^{-1}} + \frac{1}{2}\sum_{i=1}^{j}||\mathbf{y}_i - \mathbf{H}\mathbf{M}_i\mathbf{x}||^2_{\mathbf{R}^{-1}}$$

satisfies $\mathbf{M}_j\mathbf{x}_j^0 = \mathbf{x}_j^j = \mathbf{x}_j^a$ and $\mathbf{P}_j^j = \mathbf{P}_j^a$. We know that the Hessian matrix of J_j is $(\mathbf{P}_j^0)^{-1}$, allowing us to express the quadratic function J_j using its Taylor expansion around the solution $\mathbf{x}_j^0$ as

$$J_j(\mathbf{x}) = J_j(\mathbf{x}_j^0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_j^0)^{\mathrm{T}}(\mathbf{P}_j^0)^{-1}(\mathbf{x} - \mathbf{x}_j^0),$$

remembering that $\nabla J_j(\mathbf{x}_j^0) = 0$. We add the next available observation vector $\mathbf{y}_{j+1}$ to this expression and obtain the expression of the 4D-Var problem using the first $j+1$ observation vectors

$$\min_{\mathbf{x}\in\mathbb{R}^n} J_{j+1}(\mathbf{x}) = J_j(\mathbf{x}_j^0) + \frac{1}{2}||\mathbf{x} - \mathbf{x}_j^0||^2_{(\mathbf{P}_j^0)^{-1}} + \frac{1}{2}||\mathbf{y}_{j+1} - \mathbf{H}\mathbf{M}_{j+1}\mathbf{x}||^2_{\mathbf{R}^{-1}}. \quad (5.10)$$

It remains to prove that $\mathbf{x}_{j+1}^{j+1} = \mathbf{x}_{j+1}^a$ and that $\mathbf{P}_{j+1}^{j+1} = \mathbf{P}_{j+1}^a$. Observing that the first term of (5.10) is constant, we can apply the same reasoning as the one used to deduce (5.8) and (5.9) from (5.2) and write the solution of (5.10) as

$$\mathbf{x}_{j+1}^0 = \mathbf{x}_j^0 + \mathbf{P}_j^0\mathbf{M}_{j+1}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}(\mathbf{R} + \mathbf{H}\mathbf{M}_{j+1}\mathbf{P}_j^0\mathbf{M}_{j+1}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})^{-1}(\mathbf{y}_{j+1} - \mathbf{H}\mathbf{M}_{j+1}\mathbf{x}_j^0), \quad (5.11)$$

with the covariance matrix

$$\mathbf{P}_{j+1}^0 = \left((\mathbf{P}_j^0)^{-1} + \mathbf{M}_{j+1}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{M}_{j+1}\right)^{-1}. \quad (5.12)$$

If we integrate the solution $\mathbf{x}_{j+1}^0$ up to the time-step t_{j+1}, we have

$$\mathbf{x}_{j+1}^{j+1} = \mathbf{M}_{j+1}\mathbf{x}_j^0 + \mathbf{M}_{j+1}\mathbf{P}_j^0\mathbf{M}_{j+1}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}(\mathbf{R} + \mathbf{H}\mathbf{M}_{j+1}\mathbf{P}_j^0\mathbf{M}_{j+1}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})^{-1}(\mathbf{y}_{j+1} - \mathbf{H}\mathbf{M}_{j+1}\mathbf{x}_j^0).$$

Observing that

$$\begin{aligned}\mathbf{P}_j^j &= \mathrm{Cov}(\mathbf{x}_j^j, \mathbf{x}_j^j) = \mathrm{Cov}(\mathbf{M}_j\mathbf{x}_j^0, \mathbf{M}_j\mathbf{x}_j^0)\\ &= \mathbf{M}_j\mathrm{Cov}(\mathbf{x}_j^0, \mathbf{x}_j^0)\mathbf{M}_j^{\mathrm{T}} = \mathbf{M}_j\mathbf{P}_j^0\mathbf{M}_j^{\mathrm{T}},\end{aligned}$$

we can thus conclude that $\mathbf{x}_{j+1}^{j+1} = \mathbf{x}_{j+1}^a$ from Algorithm 4.1 and the recurrence assumptions, since we have

$$\begin{aligned}\mathbf{M}_{j+1}\mathbf{x}_j^0 &= \mathbf{M}_{j+1,j}\mathbf{M}_j\mathbf{x}_j^0 = \mathbf{M}_{j+1,j}\mathbf{x}_j^j\\ &= \mathbf{M}_{j+1,j}\mathbf{x}_j^a = \mathbf{x}_{j+1}^f,\end{aligned}$$

and

$$\begin{aligned}\mathbf{M}_{j+1}\mathbf{P}_j^0\mathbf{M}_{j+1}^{\mathrm{T}} &= \mathbf{M}_{j+1,j}\mathbf{M}_j\mathbf{P}_j^0\mathbf{M}_j^{\mathrm{T}}\mathbf{M}_{j+1,j}^{\mathrm{T}}\\ &= \mathbf{M}_{j+1,j}\mathbf{P}_j^j\mathbf{M}_{j+1,j}^{\mathrm{T}}\\ &= \mathbf{M}_{j+1,j}\mathbf{P}_j^a\mathbf{M}_{j+1,j}^{\mathrm{T}}\\ &= \mathbf{P}_{j+1}^f,\end{aligned} \tag{5.13}$$

and the same observation $\mathbf{y}_{j+1}$. The covariance matrix of $\mathbf{x}_{j+1}^{j+1}$ is given by

$$\begin{aligned}\mathbf{P}_{j+1}^{j+1} &= \mathrm{Cov}(\mathbf{x}_{j+1}^{j+1}, \mathbf{x}_{j+1}^{j+1})\\ &= \mathrm{Cov}(\mathbf{M}_{j+1}\mathbf{x}_{j+1}^0, \mathbf{M}_{j+1}\mathbf{x}_{j+1}^0)\\ &= \mathbf{M}_{j+1}\mathrm{Cov}(\mathbf{x}_{j+1}^0, \mathbf{x}_{j+1}^0)\mathbf{M}_{j+1}^{\mathrm{T}}\\ &= \mathbf{M}_{j+1}\mathbf{P}_{j+1}^0\mathbf{M}_{j+1}^{\mathrm{T}}.\end{aligned}$$

Replacing $\mathbf{P}_{j+1}^0$ by its expression (5.12) and using the right-hand side of (5.6) (with subscripts j and $j+1$ instead of 0 and 1, respectively), we obtain

$$\begin{aligned}\mathbf{P}_{j+1}^{j+1} &= \mathbf{M}_{j+1}\mathbf{P}_j^0\mathbf{M}_{j+1}^{\mathrm{T}}\\ &\quad - \mathbf{M}_{j+1}\mathbf{P}_j^0\mathbf{M}_{j+1}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}(\mathbf{R} + \mathbf{H}\mathbf{M}_{j+1}\mathbf{P}_j^0\mathbf{M}_{j+1}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})^{-1}\mathbf{H}\mathbf{M}_{j+1}\mathbf{P}_j^0\mathbf{M}_{j+1}.\end{aligned} \tag{5.14}$$

Using (5.13) in (5.14), we obtain $\mathbf{P}_{j+1}^{j+1} = \mathbf{P}_{j+1}^a$, which ends the proof. ∎

Note that the proof of this theorem shows (see equations (5.11) and (5.12)) how to update the solution of the 4D-Var and its covariance matrix incrementally when observations are obtained sequentially, without processing all the information from the beginning.

5.2 Connection between reduced 4D-Var / SEEK filter

In a recent paper, Krysta et al. (2011) proposed an hybridization between a reduced 4D-Var and a SEEK smoother (issued from the SEEK filter) with

the purpose to ensure the evolution of the background error covariance matrix within the reduced 4D-Var. In doing so, the authors have shown the equivalence between the reduced 4D-Var and a SEEK smoother.

In what follows, we investigate the connection between a reduced order 4D-Var and the SEEK filter presented in Algorithm 4.2. On the one hand, we define the following reduced 4D-Var problem,

$$\min_{\underline{\mathbf{x}}\in\mathbb{R}^r} \underline{J}(\underline{\mathbf{x}}) = \frac{1}{2}||\mathbf{L}_0\underline{\mathbf{x}} - \mathbf{x}^b||^2_{\mathbf{S}^{-1}} + \frac{1}{2}\sum_{i=1}^{N}||\mathbf{y}_i - \mathbf{H}_i\mathbf{M}_i\mathbf{L}_0\underline{\mathbf{x}}||^2_{\mathbf{R}_i^{-1}}, \tag{5.15}$$

where the background error covariance matrix is given by the sample covariance matrix $\mathbf{S}$ defined in (4.28) and where the basis of the reduced space is given by $\mathbf{L}_0$ which contains the r first EOFs computed from the spectral decomposition (4.29). This reduced problem (5.15) is derived using a similar approach that the one developed for the reduced incremental 4D-Var problem (3.14) but it defines a reduced state vector $\underline{\mathbf{x}}$ instead of a reduced increment $\delta\underline{\mathbf{x}}$. On the other hand, we consider the practical formulation of the SEEK filter where the initial analysis error covariance matrix, P_0^a, is based on the spectral decomposition (4.29) of $\mathbf{S}$. Note that since this matrix is symmetric, the EOFs can be chosen such that they are orthonormal. Before proving that problem (5.15) is equivalent to the SEEK filter, we have to reformulate its background term in another way. This is the purpose of the following lemma.

Lemma 5.2 *The solution of the reduced 4D-Var (5.15) is equal to the solution of*

$$\min_{\underline{\mathbf{x}}\in\mathbb{R}^r} \underline{J}(\underline{\mathbf{x}}) = \frac{1}{2}||\underline{\mathbf{x}} - \underline{\mathbf{x}}^b||^2_{\mathbf{U}_0^{-1}} + \frac{1}{2}\sum_{i=1}^{N}||\mathbf{y}_i - \mathbf{H}_i\mathbf{M}_i\mathbf{L}_0\underline{\mathbf{x}}||^2_{\mathbf{R}^{-1}}, \tag{5.16}$$

where $\underline{\mathbf{x}}^b = \mathbf{L}_0^{\mathrm{T}}\mathbf{x}^b$ *is the reduced background and* $\mathbf{U}_0$ *is the reduced background covariance matrix, which is a diagonal matrix containing the* r *largest eigenvalues of* $\mathbf{S}$.

Proof. — Since the columns of $[\mathbf{L}_0\ \hat{\mathbf{L}}_0]$ form an orthonormal basis of $\mathbb{R}^n$, we have the equality $[\mathbf{L}_0\ \hat{\mathbf{L}}_0][\mathbf{L}_0\ \hat{\mathbf{L}}_0]^{\mathrm{T}} = \mathbf{I}_n$ (see Meyer, 2000, p.320) which can be rewritten as $\mathbf{L}_0\mathbf{L}_0^{\mathrm{T}} + \hat{\mathbf{L}}_0\hat{\mathbf{L}}_0^{\mathrm{T}} = \mathbf{I}_n$ and used to decompose the background vector $\mathbf{x}^b$ as

$$\mathbf{x}^b = \mathbf{L}_0\mathbf{L}_0^{\mathrm{T}}\mathbf{x}^b + \hat{\mathbf{L}}_0\hat{\mathbf{L}}_0^{\mathrm{T}}\mathbf{x}^b. \tag{5.17}$$

Substituting (5.17) in the background term of (5.15), we obtain

$$||\mathbf{L}_0\underline{\mathbf{x}} - \mathbf{x}^b||^2_{\mathbf{S}^{-1}} = ||\mathbf{L}_0(\underline{\mathbf{x}} - \mathbf{L}_0^{\mathrm{T}}\mathbf{x}^b) - \hat{\mathbf{L}}_0\hat{\mathbf{L}}_0^{\mathrm{T}}\mathbf{x}^b||^2_{\mathbf{S}^{-1}}$$

whose right-hand side can be reformulated as

$$||\mathbf{L}_0(\underline{\mathbf{x}} - \mathbf{L}_0^{\mathrm{T}}\mathbf{x}^b)||^2_{\mathbf{S}^{-1}} + ||\hat{\mathbf{L}}_0\hat{\mathbf{L}}_0^{\mathrm{T}}\mathbf{x}^b||^2_{\mathbf{S}^{-1}} - 2(\mathbf{L}_0(\underline{\mathbf{x}} - \mathbf{L}_0^{\mathrm{T}}\mathbf{x}^b))^{\mathrm{T}}\mathbf{S}^{-1}(\hat{\mathbf{L}}_0\hat{\mathbf{L}}_0^{\mathrm{T}}\mathbf{x}^b).$$

We observe that the third term is equal to zero, using (4.22) and the decomposition of $\mathbf{S}^{-1}$ in the basis $[\mathbf{L}_0\ \hat{\mathbf{L}}_0]$

$$\mathbf{S}^{-1} = \mathbf{L}_0\mathbf{U}_0^{-1}\mathbf{L}_0^{\mathrm{T}} + \hat{\mathbf{L}}_0\hat{\mathbf{U}}_0^{-1}\hat{\mathbf{L}}_0^{\mathrm{T}}, \tag{5.18}$$

and twice the equality $\mathbf{L}_0^{\mathrm{T}}\hat{\mathbf{L}}_0 = 0$. The second term is constant and can thus be ignored for the minimization. Finally, we can reformulate the first term as

$$||\mathbf{L}_0(\underline{\mathbf{x}} - \mathbf{L}_0^{\mathrm{T}}\mathbf{x}^b)||^2_{\mathbf{S}^{-1}} = ||\underline{\mathbf{x}} - \mathbf{L}_0^{\mathrm{T}}\mathbf{x}^b||^2_{\mathbf{L}_0^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{L}_0},$$

which is equal to

$$||\underline{\mathbf{x}} - \underline{\mathbf{x}}^b||^2_{\mathbf{U}_0^{-1}}, \tag{5.19}$$

by (5.18) and where $\underline{\mathbf{x}}^b = \mathbf{L}_0^{\mathrm{T}}\mathbf{x}^b$. This concludes the proof since the background term of (5.15) is equal to (5.19) plus a constant term. ∎

We can now prove the main theoretical result of this chapter, i.e., the equivalence between the reduced 4D-Var and a specific formulations of the SEEK filter. More precisely, we consider the reduced 4D-Var (5.15) and a SEEK filter where the matrix $\mathbf{L}_0$ is build with the first r EOFs computed from the sample covariance matrix $\mathbf{S}$.

Theorem 5.3 *Suppose that the model operator is perfect and linear, that the observation operator is linear and that the same background $\mathbf{x}^b$, observations $\mathbf{y}_i$ and observation covariance matrices $\mathbf{R}_i$, are given. Assume moreover that the initial analysis of the SEEK filter $\mathbf{x}_0^a$ is equal to $\mathbf{L}_0\mathbf{L}_0^{\mathrm{T}}\mathbf{x}^b$ and that its covariance matrix $\mathbf{P}_0^a$ is equal to $\mathbf{L}_0\mathbf{U}_0\mathbf{L}_0^{\mathrm{T}}$. Then the analysis state $\mathbf{x}_i^a$ computed at time t_i by the SEEK filter and the solution produced by the reduced 4D-Var (5.15), using the first i observations, prolongated in the full space and integrated up to time t_i, are identical and have same covariance matrices. Both methods hence produce the same results at the end of the assimilation window.*

Proof. — In order to be as concise as possible, we set $\mathbf{M}_i = \mathbf{M}_{i,0}$, $\mathbf{H} = \mathbf{H}_i$ and $\mathbf{R} = \mathbf{R}_i$ throughout this proof. Generalizing it would just result in heavier notation. We have to prove that the reduced 4D-Var (5.15) is equivalent to the SEEK filter (with a specific choice for $\mathbf{x}_0^a$). Using Lemma 5.2, this amounts to proving the equivalence of this last with (5.16). To this aim, we use the same scheme as in the proof of Theorem 5.1. We first introduce the notation $\underline{\mathbf{x}}_l^0$ to denote the solution of the reduced 4D-Var (5.16) using the first l observations, and $\underline{\mathbf{P}}_l^0$ to denote the covariance matrix of $\underline{\mathbf{x}}_l^0$. Moreover, we introduce the notation $\mathbf{x}_l^k$ to denote the prolongation of the solution $\underline{\mathbf{x}}_l^0$ integrated up to time t_k and $\mathbf{P}_l^k$ to denote the covariance matrix of $\mathbf{x}_l^k$. We thus want to prove that the solution of the reduced 4D-Var (5.16) using the first l observations,

prolongated and integrated up to time t_l, is equal to the analysis produced by the SEEK filter after l iterations, i.e., $\mathbf{x}_l^l = \mathbf{x}_l^a$, and that both state vectors have the same covariance matrices, i.e., $\mathbf{P}_l^l = \mathbf{P}_l^a$, for each observation time $l = 1, \ldots, N$.

The proof is again by induction. To prove that $\mathbf{x}_1^1 = \mathbf{x}_1^a$ and $\mathbf{P}_1^1 = \mathbf{P}_1^a$, we first notice that $\mathbf{x}_0^0 = \mathbf{L}_0\underline{\mathbf{x}}_0^0 = \mathbf{L}_0\underline{\mathbf{x}}^b = \mathbf{L}_0\mathbf{L}_0^{\mathrm{T}}\mathbf{x}^b = \mathbf{x}_0^a$, since $\underline{\mathbf{x}}^b$ is the solution of the reduced 4D-Var problem using no observations and from the definition of $\underline{\mathbf{x}}^b$ in Lemma 5.2 and the assumption on $\mathbf{x}_0^a$. Therefore, the covariance matrix of $\mathbf{x}_0^0$ is given by $\mathbf{P}_0^0 = \mathbf{L}_0\mathbf{U}_0\mathbf{L}_0^{\mathrm{T}} = \mathbf{P}_0^a$. We next define the reduced 4D-Var problem when only one observation vector is available at the first time step as

$$\min_{\underline{\mathbf{x}}\in\mathbb{R}^r} \underline{J}_1(\underline{\mathbf{x}}) = \frac{1}{2}||\underline{\mathbf{x}} - \underline{\mathbf{x}}_0^0||^2_{(\underline{\mathbf{P}}_0^0)^{-1}} + \frac{1}{2}||\mathbf{y}_1 - \mathbf{H}\mathbf{M}_1\mathbf{L}_0\underline{\mathbf{x}}||^2_{\mathbf{R}^{-1}}.$$

Using the equivalent of equations (5.4), (5.5) and (5.9) from Theorem 5.1 adapted to the reduced space, we obtain as solution

$$\underline{\mathbf{x}}_1^0 = \underline{\mathbf{P}}_1^0\left(\mathbf{L}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}_1 + (\underline{\mathbf{P}}_0^0)^{-1}\underline{\mathbf{x}}_0^0\right) \tag{5.20}$$

where

$$\underline{\mathbf{P}}_1^0 = \left((\underline{\mathbf{P}}_0^0)^{-1} + \mathbf{L}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{M}_1\mathbf{L}_0\right)^{-1} \tag{5.21}$$

is its covariance matrix. The solution (5.20) can now be rewritten as

$$\begin{aligned}\underline{\mathbf{x}}_1^0 &= \underline{\mathbf{P}}_1^0\mathbf{L}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}_1 \\ &\quad + \underline{\mathbf{P}}_1^0\left[(\underline{\mathbf{P}}_0^0)^{-1} + \mathbf{L}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{M}_1\mathbf{L}_0 - \mathbf{L}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{M}_1\mathbf{L}_0\right]\underline{\mathbf{x}}_0^0,\end{aligned}$$

or, equivalently, using (5.21)

$$\underline{\mathbf{x}}_1^0 = \underline{\mathbf{x}}_0^0 + \underline{\mathbf{P}}_1^0\mathbf{L}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}_1 - \mathbf{H}\mathbf{M}_1\mathbf{L}_0\underline{\mathbf{x}}_0^0). \tag{5.22}$$

If we formulate this reduced solution in the full space $\mathbb{R}^n$ and integrate it up to the first time step, we obtain

$$\mathbf{x}_1^1 = \mathbf{M}_1\mathbf{L}_0\underline{\mathbf{x}}_1^0 = \mathbf{M}_1\mathbf{L}_0\underline{\mathbf{x}}_0^0 + \mathbf{M}_1\mathbf{L}_0\underline{\mathbf{P}}_1^0\mathbf{L}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}_1 - \mathbf{H}\mathbf{M}_1\mathbf{L}_0\underline{\mathbf{x}}_0^0),$$

and retrieve the solution of Algorithm 4.2 after one iteration. Indeed, by the algorithm, we have that $\mathbf{M}_1\mathbf{L}_0\underline{\mathbf{x}}_0^0 = \mathbf{M}_1\mathbf{x}_0^0 = \mathbf{M}_1\mathbf{x}_0^a = \mathbf{x}_1^f$ and, using (5.21) and the equality $\underline{\mathbf{P}}_0^0 = \mathbf{U}_0$, that $\underline{\mathbf{P}}_1^0 = \mathbf{U}_1$. This and the fact that we use the same observation $\mathbf{y}_1$ implies that $\mathbf{x}_1^1 = \mathbf{x}_1^a$. Moreover, the covariance matrix of $\mathbf{x}_1^1$ is given by

$$\begin{aligned}\mathbf{P}_1^1 &= \mathrm{Cov}(\mathbf{x}_1^1, \mathbf{x}_1^1) = \mathrm{Cov}(\mathbf{M}_1\mathbf{L}_0\underline{\mathbf{x}}_1^0, \mathbf{M}_1\mathbf{L}_0\underline{\mathbf{x}}_1^0) \\ &= \mathbf{M}_1\mathbf{L}_0\mathrm{Cov}(\underline{\mathbf{x}}_1^0, \underline{\mathbf{x}}_1^0)\mathbf{L}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}} \\ &= \mathbf{M}_1\mathbf{L}_0\underline{\mathbf{P}}_1^0\mathbf{L}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}} = \mathbf{L}_1\mathbf{U}_1\mathbf{L}_1^{\mathrm{T}} = \mathbf{P}_1^a,\end{aligned}$$

which is the analysis error covariance matrix given by the SEEK filter after one iteration.

Now, we suppose that when the j first observation vectors are available, the solution $\underline{\mathbf{x}}_j^0$ of the corresponding reduced 4D-Var problem

$$\min_{\underline{\mathbf{x}}\in\mathbb{R}^r} \underline{J}_j(\underline{\mathbf{x}}) = \frac{1}{2}||\underline{\mathbf{x}} - \underline{\mathbf{x}}_0^0||^2_{(\underline{\mathbf{P}}_0^0)^{-1}} + \frac{1}{2}\sum_{i=1}^{j}||\mathbf{y}_i - \mathbf{H}\mathbf{M}_i\mathbf{L}_0\underline{\mathbf{x}}||^2_{\mathbf{R}^{-1}}$$

satisfies $\mathbf{M}_j\mathbf{L}_0\underline{\mathbf{x}}_j^0 = \mathbf{x}_j^j = \mathbf{x}_j^a$ and $\mathbf{P}_j^j = \mathbf{P}_j^a$. We also assume that $\underline{\mathbf{P}}_j^0 = \mathbf{U}_j$. Since $(\underline{\mathbf{P}}_j^0)^{-1}$ is the Hessian matrix of $\underline{J}_j$, it allows us to express the quadratic function $\underline{J}_j$ using its Taylor expansion around the solution $\underline{\mathbf{x}}_j^0$ as

$$\underline{J}_j(\underline{\mathbf{x}}) = \underline{J}_j(\underline{\mathbf{x}}_j^0) + \frac{1}{2}(\underline{\mathbf{x}} - \underline{\mathbf{x}}_j^0)^{\mathrm{T}}(\underline{\mathbf{P}}_j^0)^{-1}(\underline{\mathbf{x}} - \underline{\mathbf{x}}_j^0),$$

remembering that $\nabla\underline{J}_j(\underline{\mathbf{x}}_j^0) = 0$. We add the next available observation vector $\mathbf{y}_{j+1}$ to this expression and obtain the expression of the reduced 4D-Var problem using the first $j+1$ observation vectors

$$\min_{\underline{\mathbf{x}}\in\mathbb{R}^r} \underline{J}_{j+1}(\underline{\mathbf{x}}) = \underline{J}_j(\underline{\mathbf{x}}_j^0) + \frac{1}{2}||\underline{\mathbf{x}} - \underline{\mathbf{x}}_j^0||^2_{(\underline{\mathbf{P}}_j^0)^{-1}} + \frac{1}{2}||\mathbf{y}_{j+1} - \mathbf{H}\mathbf{M}_{j+1}\mathbf{L}_0\underline{\mathbf{x}}||^2_{\mathbf{R}^{-1}}.$$

Taking into account that $\underline{J}_j(\underline{\mathbf{x}}_j^0)$ is constant, the solution of this problem, based on the same reasoning used to derive (5.21) and (5.22), may be written

$$\underline{\mathbf{x}}_{j+1}^0 = \underline{\mathbf{x}}_j^0 + \underline{\mathbf{P}}_{j+1}^0\mathbf{L}_0^{\mathrm{T}}\mathbf{M}_{j+1}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}_{j+1} - \mathbf{H}\mathbf{M}_{j+1}\mathbf{L}_0\underline{\mathbf{x}}_j^0),$$

with the covariance matrix

$$\underline{\mathbf{P}}_{j+1}^0 = \left((\underline{\mathbf{P}}_j^0)^{-1} + \mathbf{L}_0^{\mathrm{T}}\mathbf{M}_{j+1}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{M}_{j+1}\mathbf{L}_0\right)^{-1}.$$

By Algorithm 4.2, we remark that $\underline{\mathbf{P}}_{j+1}^0 = \mathbf{U}_{j+1}$ since $\underline{\mathbf{P}}_j^0 = \mathbf{U}_j$. If we formulate the solution $\underline{\mathbf{x}}_{j+1}^0$ in the full space and integrate it up to the time-step $j+1$, we have

$$\begin{aligned}\mathbf{x}_{j+1}^{j+1} &= \mathbf{M}_{j+1}\mathbf{L}_0\underline{\mathbf{x}}_{j+1}^0 \\ &= \mathbf{M}_{j+1}\mathbf{L}_0\underline{\mathbf{x}}_j^0 + \mathbf{M}_{j+1}\mathbf{L}_0\underline{\mathbf{P}}_{j+1}^0\mathbf{L}_0^{\mathrm{T}}\mathbf{M}_{j+1}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{y}_{j+1} - \mathbf{H}\mathbf{M}_{j+1}\mathbf{L}_0\underline{\mathbf{x}}_j^0).\end{aligned}$$

Using the recurrence assumptions, we retrieve the SEEK filter solution $\mathbf{x}_{j+1}^a$ and its covariance matrix after $j+1$ iterations, since $\underline{\mathbf{P}}_{j+1}^0 = \mathbf{U}_{j+1}$, and

$$\begin{aligned}\mathbf{M}_{j+1}\mathbf{L}_0\underline{\mathbf{x}}_j^0 &= \mathbf{M}_{j+1,j}\mathbf{M}_j\mathbf{L}_0\underline{\mathbf{x}}_j^0 = \mathbf{M}_{j+1,j}\mathbf{x}_j^j \\ &= \mathbf{M}_{j+1,j}\mathbf{x}_j^a = \mathbf{x}_{j+1}^f\end{aligned}$$

and

$$\begin{aligned}
\mathbf{P}_{j+1}^{j+1} &= \mathrm{Cov}(\mathbf{x}_{j+1}^{j+1}, \mathbf{x}_{j+1}^{j+1}) \\
&= \mathrm{Cov}(\mathbf{M}_{j+1}\mathbf{L}_0\underline{\mathbf{x}}_{j+1}^0, \mathbf{M}_{j+1}\mathbf{L}_0\underline{\mathbf{x}}_{j+1}^0) \\
&= \mathbf{M}_{j+1}\mathbf{L}_0\mathrm{Cov}(\underline{\mathbf{x}}_{j+1}^0, \underline{\mathbf{x}}_{j+1}^0)\mathbf{L}_0^{\mathrm{T}}\mathbf{M}_{j+1}^{\mathrm{T}} \\
&= \mathbf{M}_{j+1}\mathbf{L}_0\underline{\mathbf{P}}_{j+1}^0\mathbf{L}_0^{\mathrm{T}}\mathbf{M}_{j+1}^{\mathrm{T}} \\
&= \mathbf{L}_{j+1}\mathbf{U}_{j+1}\mathbf{L}_{j+1}^{\mathrm{T}} \\
&= \mathbf{P}_{j+1}^a
\end{aligned}$$

by Algorithm 4.2. This ends the proof. ∎

The two theorems presented in this chapter state connections between the variational and the sequential approaches in the specific case of linear observation and model operators. However, these theorems have also a usefulness in the case of nonlinear operators. Recalling that the nonlinear 4D-Var problems are solved by a sequence of incremental 4D-Var problems (see Section 3.2), the theorems can be applied to the incremental 4D-Var problems since they use linear operators (Jacobians of the nonlinear ones). As an example, the reduced incremental 4D-Var problem (3.14) can be reformulated as

$$\min_{\delta\underline{\mathbf{x}}\in\mathbb{R}^r} ||\mathbf{L}\delta\underline{\mathbf{x}} - \mathbf{d}^{bk}||^2_{\mathbf{B}^{-1}} + \frac{1}{2}\sum_{i=1}^{N} ||(\mathbf{y}_i - \mathcal{H}_i\mathcal{M}_{i,0}\mathbf{x}_k) - \mathbf{H}_i^k\mathbf{M}_i^k\mathbf{L}\delta\underline{\mathbf{x}}||^2_{\mathbf{R}_i^{-1}}, \quad (5.23)$$

using the weighted norms and the definitions (3.7) and (3.11). If the background error covariance matrix $\mathbf{B}$ is replaced by the sample covariance matrix $\mathbf{S}$ and if the reduced space $\mathbf{L}$ contains the first r EOFs, this problem has the same form as (5.15), equalizing

$$\begin{aligned}
\underline{\mathbf{x}} &:= \delta\underline{\mathbf{x}} \\
\mathbf{x}^b &:= \mathbf{d}^{bk} \\
\mathbf{y}_i &:= \mathbf{y}_i - \mathcal{H}_i\mathcal{M}_{i,0}\mathbf{x}_k \\
\mathbf{H}_i &:= \mathbf{H}_i^k \\
\mathbf{M}_i &:= \mathbf{M}_i^k.
\end{aligned}$$

and remarking that $\mathbf{L}$ is given by $\mathbf{L}_0$ from (4.29). Thus, the Theorem 5.3 can be applied to (5.23) and proves the equivalence between its solution and a specific SEEK filter.

Chapter 6

Minimizing the 4D-Var problem

The 4D-var problem has been presented in Chapter 3 where a first rough idea on its solution has been given with the incremental approach. In this chapter, we give a survey on the basic optimization methods designed for solving nonlinear problems with continuous variables. It will allow us to highlight the connection between the incremental approach and the Gauss-Newton method since they both compute the solution of the 4D-Var problem by solving the same sequence of symmetric positive definite linear systems. Two suited methods for solving such linear systems, namely the conjugate gradient method and the Lanczos method are presented and analyzed. Their rate of convergence depends, among other things, on the condition number of the matrix and on the starting point. The remainder of this chapter is then devoted to present a limited-memory preconditioner with an appropriate starting point, both based on information from EOFs, in an attempt to accelerate the Gauss-Newton method further. Numerical experiments performed on a shallow water model are presented at the end of the chapter. The behavior of this approach is also illustrate with the NEMO framework in Chapter 8.

6.1 Unconstrained nonlinear optimization

Unconstrained nonlinear optimization problems attend to find a vector $\mathbf{x}$ that minimizes an objective function

$$f : \mathbb{R}^n \to \mathbb{R} : \mathbf{x} \rightsquigarrow f(\mathbf{x}),$$

without restriction on the values of $\mathbf{x}$. Mathematically, the problem is formulated as

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}).$$

The objective function f is assumed to be smooth, which signifies that the derivatives of all orders are available. In most optimization problems, the graph of the function is unknown and it is only possible to evaluate the function and its first, possibly second, order derivative on particular points. Since these evaluations have a computational cost, the optimization algorithm has to find a solution performing a minimal number of evaluations. Because the objective function f is nonlinear and there is no assumption about its convexity, it is possible that the objective function has multiple local minimizers with a global one. Of course, it would be preferable to have an optimization algorithm which finds the global minimizer. However, this task is tough and there are no efficient algorithms which guarantee to find a global minimizer for any problem (Ugray et al., 2007). Most of the optimization algorithms are iterative, computing a sequence of points $\{\mathbf{x}_k\}$ which converges to a first-order critical point, i.e., a point where the norm of the gradient of f is null. At each iteration k, the optimization algorithm computes a new iterate $\mathbf{x}_{k+1}$ with a lower function value[1] than at $\mathbf{x}_k$ using the information from the previous iterations and from the evaluation of f and its derivatives at $\mathbf{x}_k$. There are two fundamental strategies for computing such a sequence: the *line search* and the *trust-region* strategies. We review briefly these strategies in what follows.

In the line search strategy (Griva et al., 2009, Nocedal and Wright, 2006), the algorithm computes, at each iteration k, a direction $\mathbf{p}_k$ and searches along this direction for a point which sufficiently reduces the function value. More precisely, a step length α_k is computed by approximately solving

$$\min_{\alpha\in\mathbb{R}^+} f(\mathbf{x}_k + \alpha\mathbf{p}_k), \tag{6.1}$$

and the new iterate is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k.$$

Of course, solving (6.1) exactly produces an iterate with the maximal decrease along the direction $\mathbf{p}_k$. Unfortunately, it may be expensive to compute since it may involve a lot of objective function evaluations. Instead, the line search algorithm generates only a limited number of trial steps and stops generally once a sufficient curvature and a decrease condition is obtained. To this aim, various conditions such as the Wolfe conditions or the Goldstein conditions can be used (Nocedal and Wright, 2006). The search direction $\mathbf{p}_k$ can be computed in different ways. The most obvious one is $\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$ since it is the direction where

[1] The family of non-monotone optimization method relax this constraint and the sequence $\{f(\mathbf{x}_k)\}$ is no longer monotonically decreasing.

the function decreases the most rapidly. The line search method which uses this direction as search direction is called the *steepest descent method*. However, this method may converge slowly, suffering of oscillatory effects in the convergence since it uses only the first-order derivative. The Newton direction is the most famous search direction and is defined as $\mathbf{p}_k = -\left(\nabla^2 f\left(\mathbf{x}_k\right)\right)^{-1} \nabla f(\mathbf{x}_k)$. It uses derivative information up to the second-order. Its derivation comes from the fact that this direction, with a step length of one, minimizes the second-order Taylor expansion of f around $\mathbf{x}_k$. The line search method which uses Newton direction as search direction is called the *Newton method*. Some caution must be made with the use of this direction. Indeed, if the Hessian matrix $\nabla^2 f(\mathbf{x}_k)$ is not positive definite, the inverse may not exists or the search direction may not be a descent direction, i.e., $\nabla f(\mathbf{x}_k)^T p_k < 0$. In these cases, some modification of the Newton direction must be performed (Nocedal and Wright, 2006).

The trust-region strategy (Conn et al., 2000) uses a rather different approach than the line search strategy. At each iteration k, a model m_k of the objective function is constructed around the current iterate $\mathbf{x}_k$ using the information from the function value and its derivatives at $\mathbf{x}_k$. Usually, a quadratic model is used

$$m_k(\mathbf{x}_k + \mathbf{p}) = f(\mathbf{x}_k) + \mathbf{p}^T \nabla f(\mathbf{x}_k) + \frac{1}{2}\mathbf{p}^T \nabla^2 f(\mathbf{x}_k)\mathbf{p}.$$

Since the model is a good surrogate of the objective function only in a local region around $\mathbf{x}_k$, it is minimized in such a region, called the trust region and defined by a trust-region radius Δ_k,

$$\min_{||\mathbf{p}|| \leqslant \Delta_k} m_k(\mathbf{x}_k + \mathbf{p}),$$

using usually the Euclidean norm. On the one hand, if the decrease in the model yields a decrease in the objective function, the trial point $\mathbf{x}_k + \mathbf{p}_k$ is accepted and the new iterate is given by $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$. The trust region could be increased for the next iteration since it seems that the model represents well the objective function. On the other hand, if the decrease in the model yields an increase in the objective function, the trial point $\mathbf{x}_k + \mathbf{p}_k$ is rejected and the new iterate is given by $\mathbf{x}_{k+1} = \mathbf{x}_k$. The trust region is shrunk, hoping that the model will better represent the function value in a smaller region.

Optimization algorithms using second-order derivatives perform better than the ones using only first-order derivatives. However computing second-order derivatives can be an expensive process. Finite differences or automatic differentiation can be used to avoid hand writing but the evaluation can remain too costly. A possible solution, for both line search an trust-region strategies, is to use *quasi-Newton* methods. These methods replace the exact Hessians by approximations updated at each iteration from previous first-order information. The most famous rules are the symmetric-rank-one (SR1) formula and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula. The BFGS formula

updates an inverse Hessian approximation $\mathbf{H}_k$[2] after each iteration of the line search or the trust-region method. Assuming that the iterate $\mathbf{x}_k$ has been computed, the BFGS formula computes $\mathbf{H}_k$ using the previous approximation $\mathbf{H}_{k-1}$ and the two last iterates $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$ with their gradient evaluations $\nabla f(\mathbf{x}_k)$ and $\nabla f(\mathbf{x}_{k-1})$, as

$$\mathbf{H}_k = \left(\mathbf{I}_n - \frac{\mathbf{s}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{s}_k}\right)\mathbf{H}_{k-1}\left(\mathbf{I}_n - \frac{\mathbf{y}_k\mathbf{s}_k^T}{\mathbf{y}_k^T\mathbf{s}_k}\right) + \frac{\mathbf{s}_k\mathbf{s}_k^T}{\mathbf{y}_k^T\mathbf{s}_k}, \tag{6.2}$$

with $\mathbf{s}_k = \mathbf{x}_k - \mathbf{x}_{k-1}$ and $\mathbf{y}_k = \nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_{k-1})$. The BFGS formula produces symmetric positive definite Hessian approximations (ensuring that the Newton directions are descent directions) whenever the initial approximation $\mathbf{H}_0$ is symmetric positive definite and $\mathbf{s}_k\mathbf{y}_k > 0$ (see, Nocedal and Wright, 2006, Chapter 6).

The line search and the trust-region strategies are globalization techniques. It means that, using some basic assumptions, both methods converge (theoretically) to first-order critical points, independently of the chosen starting point $\mathbf{x}_0$[3](Nocedal and Wright, 2006). However, this nice convergence result may suffer from numerical instabilities which may slow down the convergence rate or even break the convergence.

Some optimization problems have special forms which can be exploited to facilitate the minimization. Amongst others, nonlinear least squares problems are defined as

$$\min_{\mathbf{x}\in\mathbb{R}^n} \; f(\mathbf{x}) = \frac{1}{2}||\mathbf{r}(\mathbf{x})||_2^2, \tag{6.3}$$

where

$$\mathbf{r} : \mathbb{R}^n \to \mathbb{R}^m : \mathbf{x} \rightsquigarrow \mathbf{r}(\mathbf{x}) = (\mathbf{r}_1(\mathbf{x}), \ldots, \mathbf{r}_m(\mathbf{x}))^T$$

is a residual function which is assumed to be smooth (Bjorck, 1996). These problems arise frequently in chemistry, physics or economy when a model must be parameterized using observations. In our work, we are concerned with nonlinear least squares problems since the 4D-Var problem belongs to this class. Before presenting a special formulation of the line search and trust-region strategy devoted for nonlinear least squares problems, it is useful to express the first and second derivatives of f in (6.3) in terms of the Jacobian of $\mathbf{r}$, denoted by $\mathbf{J}$. The gradient of f is given by

$$\begin{aligned}\nabla f(\mathbf{x}) &= \sum_{i=1}^{m} \mathbf{r}_i(\mathbf{x})\nabla\mathbf{r}_i(\mathbf{x}) \\ &= \mathbf{J}(\mathbf{x})^T\mathbf{r}(\mathbf{x}).\end{aligned} \tag{6.4}$$

[2] Be careful not to confuse this Hessian approximation with the observation operator.

[3] Pay attention that the word "globalization" does not mean that the method converges to a global minimizer.

and the Hessian by

$$\nabla^2 f(\mathbf{x}) = \sum_{i=1}^{m} \nabla \mathbf{r}_i(\mathbf{x}) \nabla \mathbf{r}_i(\mathbf{x})^T + \sum_{i=1}^{m} \mathbf{r}_i(\mathbf{x}) \nabla^2 \mathbf{r}_i(\mathbf{x})$$
$$= \mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}) + \sum_{i=1}^{m} \mathbf{r}_i(\mathbf{x}) \nabla^2 \mathbf{r}_i(\mathbf{x}). \tag{6.5}$$

Most of the algorithms for nonlinear least squares problems will approximate the Hessian neglecting the second term in (6.5). This assumption is commonly made since one can hope that the first term usually dominate the second ones. It allows to compute an approximation of the Hessian only using first-order derivatives of $\mathbf{r}_i$:

$$\nabla^2 f(\mathbf{x}) \approx \mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}). \tag{6.6}$$

The *Gauss-Newton* method is a line search algorithm which solves nonlinear least squares problems using the approximation (6.6) (Sun and Yuan, 2006). It belongs to the family of quasi-Newton methods since the exact Hessian is approximated. At each iteration, the search direction is given by the solution of the following linear system

$$\mathbf{J}(\mathbf{x}_k)^T \mathbf{J}(\mathbf{x}_k) \mathbf{p} = -\mathbf{J}(\mathbf{x}_k)^T \mathbf{r}(\mathbf{x}_k), \tag{6.7}$$

known as the *normal equations*. The system (6.7) can be rewritten in a more compact form as

$$\mathbf{A}_k \mathbf{p} = \mathbf{b}_k, \tag{6.8}$$

where $\mathbf{A}_k = \mathbf{J}(\mathbf{x}_k)^T \mathbf{J}(\mathbf{x}_k)$ and $\mathbf{b}_k = -\mathbf{J}(\mathbf{x}_k)^T \mathbf{r}(\mathbf{x}_k)$. If the Jacobian has full rank, the matrix $\mathbf{A}_k$ is symmetric definite positive and the search direction is a descent direction. Symmetric positive definite linear systems like (6.8) can be solved using direct methods based on the Cholesky factorization of $\mathbf{A}_k$ or on the QR factorization of $\mathbf{J}_k$ (Davis, 2006). However, iterative methods, such as the *conjugate gradient* and the *Lanczos* method, are better suited for large systems since they do not require the explicit storage of $\mathbf{A}_k$ but only a routine which computes matrix-vector products. Once the descent direction is computed, a line search is performed as previously.

The Levenberg-Marquardt method is a trust region-method using the approximation (6.6) (Sun and Yuan, 2006). At each iteration the following model is minimized in a trust region

$$\min_{||\mathbf{p}|| \leqslant \Delta_k} \frac{1}{2} ||\mathbf{r}(\mathbf{x}_k)||_2^2 + \mathbf{p}^T \mathbf{J}(\mathbf{x}_k)^T \mathbf{r}(\mathbf{x}_k) + \frac{1}{2} \mathbf{p}^T \mathbf{J}(x_k)^T \mathbf{J}(\mathbf{x}_k) \mathbf{p},$$

which is a quadratic model where (6.4) and (6.6) are used for the first and second derivatives, respectively. This problem is equivalent to the minimization of

$$\min_{||\mathbf{p}|| \leqslant \Delta_k} \frac{1}{2} ||\mathbf{J}(\mathbf{x}_k) \mathbf{p} + \mathbf{r}(\mathbf{x}_k)||_2^2,$$

which belongs to the class of linear least-squares problems. Since the Gauss-Newton and the Levenberg-Marquardt methods use the line search or the trust region framework, they have the nice property of convergence to first-order critical points (Nocedal and Wright, 2006). In the next section, we show that the incremental method presented in Chapter 3 is related to a modified Gauss-Newton method.

6.2 Gauss-Newton method for the 4D-Var problem

In atmospheric and oceanographic data assimilation problems, the model and observation operators are usually nonlinear and the 4D-Var problem (3.6) belongs to the class of nonlinear least-squares problems. Indeed, it can be rewritten in the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} J^{\mathsf{4D}}(\mathbf{x}) = \frac{1}{2}||\mathbf{r}(\mathbf{x})||_2^2, \tag{6.9}$$

with

$$\mathbf{r}(\mathbf{x}) = \begin{pmatrix} \mathbf{B}^{-1/2}(\mathbf{x}-\mathbf{x}^b) \\ \mathbf{R}^{-1/2}(\mathcal{G}(\mathbf{x})-\mathbf{y}) \end{pmatrix}, \tag{6.10}$$

where the matrices $\mathbf{B}^{-1/2}$ and $\mathbf{R}^{-1/2}$ are the inverses of the Cholesky factors given by $\mathbf{B} = \mathbf{B}^{1/2}(\mathbf{B}^{1/2})^T$ and $\mathbf{R} = \mathbf{R}^{1/2}(\mathbf{R}^{1/2})^T$, respectively. The Gauss-Newton method solves a sequence of linear systems (6.7) to compute the search directions. For the 4D-Var problem, the Jacobian matrix of $\mathbf{r}$ at $\mathbf{x}_k$ is given by

$$\mathbf{J}(\mathbf{x}_k) = \begin{pmatrix} \mathbf{B}^{-1/2} \\ \mathbf{R}^{-1/2}\mathbf{G}_k \end{pmatrix}, \tag{6.11}$$

where $\mathbf{G}_k$ is the Jacobian matrix of $\mathcal{G}$ at $\mathbf{x}_k$ defined by (3.7). The matrix and the right hand side of the system (6.8) are given by

$$\mathbf{A}_k = \mathbf{J}(\mathbf{x}_k)^T\mathbf{J}(\mathbf{x}_k) = \mathbf{B}^{-1} + \mathbf{G}_k^T\mathbf{R}^{-1}\mathbf{G}_k, \tag{6.12}$$

and

$$\begin{aligned} \mathbf{b}_k &= -\mathbf{J}(\mathbf{x}_k)^T\mathbf{r}(\mathbf{x}_k) \\ &= \mathbf{B}^{-1}(\mathbf{x}^b - \mathbf{x}_k) + \mathbf{G}_k^T\mathbf{R}^{-1}(\mathbf{y} - \mathcal{G}(\mathbf{x}_k)) \\ &= \mathbf{B}^{-1}\mathbf{d}^{bk} + \mathbf{G}_k^T\mathbf{R}^{-1}\mathbf{d}^{ok} \end{aligned} \tag{6.13}$$

using (3.10) and (3.11). The matrix $\mathbf{A}_k$ is a symmetric positive definite matrix which ensures that the solution of the linear system, $\mathbf{p}_k$, is a descent direction. One can remark that the solution of the normal equations (6.7) is equivalent to the minimization of

$$\min_{\mathbf{p}\in\mathbb{R}^n} \frac{1}{2}||\mathbf{J}(\mathbf{x}_k)\mathbf{p} + \mathbf{r}(\mathbf{x}_k)||_2^2 \tag{6.14}$$

by nullifying its gradient, using the formula (A.7). This linear least squares problem can be expanded using (6.10) and (6.11) as

$$\min_{\mathbf{p}\in\mathbb{R}^n} \frac{1}{2}\left(\mathbf{p}-(\mathbf{x}^b-\mathbf{x}_k)\right)^T \mathbf{B}^{-1}\left(\delta\mathbf{x}-(\mathbf{x}^b-\mathbf{x}_k)\right) + \frac{1}{2}\left(\mathbf{G}_k\mathbf{p}-(\mathbf{y}-\mathcal{G}(\mathbf{x}_k))\right)^T \mathbf{R}^{-1}\left(\mathbf{G}_k\mathbf{p}-(\mathbf{y}-\mathcal{G}(\mathbf{x}_k))\right). \quad (6.15)$$

Performing the change of variable $\delta\mathbf{x} = \mathbf{p}$ and using the definition of the departure vectors given by (3.10) and (3.11), we obtain

$$\min_{\delta\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}(\delta\mathbf{x}-\mathbf{d}^{bk})^T\mathbf{B}^{-1}(\delta\mathbf{x}-\mathbf{d}^{bk}) + \frac{1}{2}(\mathbf{G}_k\delta\mathbf{x}-\mathbf{d}^{ok})^T\mathbf{R}^{-1}(\mathbf{G}_k\delta\mathbf{x}-\mathbf{d}^{ok}), \quad (6.16)$$

and we recover the incremental 4D-Var formulation $J^{\texttt{4D-inc}}(\delta\mathbf{x})$ introduced in (3.8). It shows that the sequence of linear systems defined in the Gauss-Newton method is equivalent to the sequence of incremental problems.

For operational weather forecast, the computational budget is very limited since the forecast must be broadcast before being obsolete. For this reason, the sequence of incremental problems has to be shorten (around 2 to 5 problems are stated) and each incremental problem has to be solved approximately (using an iterative method and performing around 10 iterations) (Weaver et al., 2003). With this computational constraint, it is not conceivable to expect the convergence to a first-order critical point and we can just hope to decrease the 4D-Var function as much as possible. One can remark that the incremental approach is thus equivalent to the Gauss-Newton method where only few approximative descent directions are computed without line search along them (Lawless et al., 2004, Gratton et al., 2007). We summarize the Gauss-Newton method for solving the 4D-Var in Algorithm 6.1.

In the next two subsections, we present the conjugate-gradient and the Lanczos methods which aim to solve the normal equations (6.7) or, equivalently, to minimize the incremental problem (6.16). Both methods are iterative and suited for solving large problems since they do not require explicit matrix storage but only matrix-vector products.

6.2.1 The conjugate gradient method

The conjugate gradient (CG) method has been introduced by Hestenes and Stiefel (1952) to solve symmetric positive definite linear systems and belongs to the class of Krylov subspace methods (Meurant and Strakos, 2006). Its philosophy can be understood using the equivalence between the solution of a symmetric positive definite linear system,

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

Algorithm 6.1 Gauss-Newton algorithm for the 4D-Var problem

1: Define an initial guess at time t_0, usually, $\mathbf{x}_0 = \mathbf{x}^b$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Run the nonlinear model from the current iterate $\mathbf{x}_k$ to update the reference trajectory and store the vector $\mathcal{G}(\mathbf{x}_k)$.
4: Calculate the departure vectors $\mathbf{d}^{bk}$ and $\mathbf{d}^{ok}$
5: Approximately solve

$$\min_{\delta\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}(\delta\mathbf{x} - \mathbf{d}^{bk})^T\mathbf{B}^{-1}(\delta\mathbf{x} - \mathbf{d}^{bk}) + \frac{1}{2}(\mathbf{G}_k\delta\mathbf{x} - \mathbf{d}^{ok})^T\mathbf{R}^{-1}(\mathbf{G}_k\delta\mathbf{x} - \mathbf{d}^{ok}),$$

or equivalently

$$\mathbf{A}_k\delta\mathbf{x} = \mathbf{b}_k$$

with $\mathbf{A}_k$ and $\mathbf{b}_k$ given by (6.12) and (6.13).
6: Update the iterate with the increment $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta\mathbf{x}_k$
7: **end for**

and the minimization of a strictly convex quadratic function

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}. \tag{6.17}$$

The CG algorithm is a special line search method which constructs a sequence of $\mathbf{A}$-conjugate directions $\{\mathbf{s}_j\}$, i.e.,

$$\mathbf{s}_i\mathbf{A}\mathbf{s}_j = 0, \qquad \forall\, i \neq j,$$

as search directions. The step length performed along each direction is the result of an exact minimization of the quadratic function along this direction. The importance of $\mathbf{A}$-conjugacy lies in the fact that the quadratic function can be minimized in n steps (for $\mathbf{A} \in \mathbb{R}^{n\times n}$) by successively minimizing along each $\mathbf{A}$-conjugate direction (see Nocedal and Wright, 2006, p.103, for the proof). The CG method is summarized in Algorithm 6.2. Its computational cost is mainly dominated by the matrix-vector product between $\mathbf{A}$ and $\mathbf{s}_j$ performed at each iteration. Some interesting properties are summarized in the next theorem.

Theorem 6.1 *Suppose that the jth iterate generated by the conjugate gradient method is not the solution. The following properties hold*

1. $\mathbf{x}_j$ *is the minimizer of the quadratic function over the set* $\{\mathbf{x} \mid \mathbf{x} = \mathbf{x}_0 + span\,\{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{j-1}\}\}$,
2. $span\,\{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{j-1}\} = \mathcal{K}(\mathbf{r}_0, j)$,

Algorithm 6.2 Conjugate gradient method to solve $\mathbf{Ax} = \mathbf{b}$

$j = 0$
Given $\mathbf{x}_0$
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
for $j = 0, 1, 2, \ldots$ **do**
  if $j = 0$ **then**
    $\mathbf{s}_0 = \mathbf{r}_0$
  else
    $\beta_j = \mathbf{r}_j^T \mathbf{r}_j / \mathbf{r}_{j-1}^T \mathbf{r}_{j-1}$
    $\mathbf{s}_j = \mathbf{r}_j + \beta_j \mathbf{s}_{j-1}$
  end if
  $\alpha_j = \mathbf{r}_j^T \mathbf{s}_j / \mathbf{s}_j^T \mathbf{A}\mathbf{s}_j$
  $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{s}_j$
  $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A}\mathbf{s}_j$
  $j = j + 1$
end for

3. *span* $\{\mathbf{r}_0, r_1, \ldots, \mathbf{r}_{j-1}\} = \mathcal{K}(\mathbf{r}_0, j)$,

where $\mathcal{K}(\mathbf{r}_0, j)$ *is the Krylov subspace of degree* j *and is defined as span* $\left\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{j-1}\mathbf{r}_0\right\}$.

Proof. — (Nocedal and Wright, 2006, p.106 and p.109) ∎

This result shows that the CG method minimizes the quadratic function in nested Krylov subspaces of increasing dimension. Its rate of convergence depends mostly on the eigenvalues distribution of the matrix. There are two main results which are expressed in the next two theorems.

Theorem 6.2 *If* $\mathbf{A}$ *has only* r *distinct eigenvalues, then the CG iteration will terminate at the solution in at most* r *iterations.*

Proof. — (Nocedal and Wright, 2006, p.115) ∎

Theorem 6.3

$$||\mathbf{x}_j - \mathbf{x}^*||_{\mathbf{A}} \leqslant 2 \left(\frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1} \right)^j ||\mathbf{x}_0 - \mathbf{x}^*||_{\mathbf{A}},$$

where $\kappa(\mathbf{A}) = \lambda_n / \lambda_1$ *is the ration between the largest and the smallest eigenvalues of* $\mathbf{A}$*, called the condition number of* $\mathbf{A}$.

Proof. — (Conn et al., 2000, p.82) ∎

These results show that the rate of convergence depends on the clusterization of the eigenvalues and on the condition number $\kappa(\mathbf{A})$ of the system matrix $\mathbf{A}$. The larger is $\kappa(\mathbf{A})$, the slower the convergence. The CG method can theoretically be viewed as a direct method, as it produces, in exact arithmetic, the solution after a finite number of iterations, which is not larger than n. However, it suffers of a lack of robustness and efficiency. In practice, the directions are not $\mathbf{A}$-conjugate, there are instabilities when $\mathbf{s}_j^T \mathbf{A}\mathbf{s}_j$ is small and the convergence can be extremely slower than the theoretical rate (Meurant and Strakos, 2006). Fortunately, the CG method can be used as an iterative method as it decreases monotonically the quadratic function. An effective stopping criteria is to stop the algorithm when the norm of the residual, $||\mathbf{r}_j||$, is reduced to a user-specified tolerance.

Both the efficiency and the robustness of the CG method can be improved by using *preconditioning techniques*. Preconditioning is a simple mean of transforming the original linear system into another one which has the same solution, but which is likely to be easier to solve with an iterative method (Saad, 2008). It is widely recognized that there is no special and universal way to design a preconditioner for all types of problems (Benzi, 2002). But ideally, a preconditioner must:

- be an approximation of $\mathbf{A}^{-1}$ (inverse type) or be the inverse of an approximation of $\mathbf{A}$ (forward type);
- be symmetric and positive definite;
- be cheap to apply;
- reduce the condition number and/or cluster the eigenvalues.

There are in general two approaches to construct a preconditioner. On the one hand, physical-based preconditioners require the knowledge of the application and are problem dependent. The most famous ones use geometric multigrid or domain decomposition algorithms (Chen, 2005). On the other hand, algebraic preconditioners use only information contained in the entrances of $\mathbf{A}$. They achieve reasonable efficiency on a wide range of problems. As an example, the incomplete Cholesky factorization (a sparse approximation of the Cholesky factorization) can be used as a preconditioner (Manteuffel, 1980).

Assuming an inverse preconditioner $\mathbf{P}$, there are three main different ways of applying it. To the left, it leads to the following preconditioned system

$$\mathbf{PAx} = \mathbf{Pb}.$$

To the right, we obtain

$$\mathbf{APy} = \mathbf{b},$$

where the solution is computed by performing a change of variables $\mathbf{x} = \mathbf{P}\mathbf{y}$. The third possible way is to compute a preconditioner in a factored form,

$$\mathbf{P} = \mathbf{S}\mathbf{S}^T,$$

and to apply it in a splitting form, giving the following preconditioned system,

$$\mathbf{S}^T\mathbf{A}\mathbf{S}\mathbf{y} = \mathbf{S}^T\mathbf{b}.$$

where the solution is given by $\mathbf{x} = \mathbf{S}\mathbf{y}$. The third way is the only one which preserves the symmetry of the original linear system.

The CG algorithm can be reformulated when a preconditioner is used. Remembering that the CG method solves only symmetric positive definite linear systems, the split preconditioner seems to be the best suited. It leads to the split preconditioned CG method presented in Algorithm 6.3. However, there

Algorithm 6.3 Preconditioned Conjugate gradient to solve $\mathbf{A}\mathbf{x} = \mathbf{b}$

$j = 0$
Given $\mathbf{x}_0$
$\mathbf{r}_0 = \mathbf{S}^T(\mathbf{b} - \mathbf{A}\mathbf{x}_0)$
for $j = 0, 1, 2, \ldots$ **do**
  if $j = 0$ **then**
    $\mathbf{p}_0 = \mathbf{S}\mathbf{r}_0$
  else
    $\beta_j = \mathbf{r}_j^T\mathbf{r}_j / \mathbf{r}_{j-1}^T\mathbf{r}_{j-1}$
    $\mathbf{p}_j = \mathbf{S}\mathbf{r}_j + \beta_j\mathbf{p}_{j-1}$
  end if
  $\alpha_j = \mathbf{r}_j^T\mathbf{p}_j / \mathbf{p}_j^T\mathbf{A}\mathbf{p}_j$
  $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j\mathbf{p}_j$
  $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j\mathbf{S}^T\mathbf{A}\mathbf{p}_j$
  $j = j + 1$
end for

is an alternative of the split preconditioner for the CG method which does not require the split form. It replaces the usual Euclidean inner product by the $\mathbf{P}$-inner product, where $\mathbf{P} = \mathbf{S}^T\mathbf{S}$ (see Golub and Van Loan, 1996, p.534). No approach is better than the other since the produced iterates are identical (see Saad, 2008, p.278).

6.2.2 The Lanczos method

The Lanczos method is usually thought as a method for constructing an orthonormal basis of a Krylov subspace or for estimating eigenvalues and eigenvectors of a symmetric matrix (Stewart, 2001). However, it can be used to solve symmetric indefinite linear systems. The Lanczos method is summarized

Algorithm 6.4 Lanczos method

$j = 0$
Given $\mathbf{r}_0$
$\mathbf{y}_0 = \mathbf{r}_0$
$\mathbf{q}_{-1} = 0$
for $j = 0, 1, 2, \ldots$ **do**
  $\gamma_j = ||\mathbf{y}_0||_2$
  $\mathbf{q}_j = \mathbf{y}_j / \gamma_j$
  $\delta_j = \mathbf{q}_j^T \mathbf{A}\mathbf{q}_j$
  $\mathbf{y}_{j+1} = \mathbf{A}\mathbf{q}_j - \delta_j \mathbf{q}_j - \gamma_j \mathbf{q}_{j-1}$
  $j = j + 1$
end for

in Algorithm 6.4. Its goal is to produce a sequence of Lanczos vectors $\{\mathbf{q}_j\}$ which form an orthonormal basis of $\mathcal{K}(\mathbf{r}_0, j)$. Rounding errors may effect the behavior of the Lanczos algorithm. These are mainly due to the loss of orthogonality between the Lanczos vectors. To restore the orthogonality in the Lanczos vectors, it is recommended to orthogonalize each Lanczos vector $\mathbf{q}_j$ with respect to those of the preceding iterations. This process can be done using a Gram-Schmidt algorithm (Golub and Van Loan, 1996, p.230 and p.482). The Lanczos vectors can be used to solve the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ thanks to the following result.

Theorem 6.4 *Assume that* $\mathbf{Q}_j = [\mathbf{q}_0, \ldots, \mathbf{q}_{j-1}] \in \mathbb{R}^{n \times j}$ *is the matrix of Lanczos vectors obtained after* j *iterations with* $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ *for a given* $\mathbf{x}_0$*. Then the vector*

$$\mathbf{x}_j = \mathbf{x}_0 + \mathbf{Q}_j \mathbf{w}_j,$$

where $\mathbf{w}_j$ *is the solution of*

$$(\mathbf{Q}_j^T \mathbf{A}\mathbf{Q}_j)\mathbf{w} = \mathbf{Q}_j^T \mathbf{r}_0, \tag{6.18}$$

minimizes the quadratic function (6.17) over the set $\{\mathbf{x} \mid \mathbf{x} = \mathbf{x}_0 + span\,\{\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_{j-1}\}\}$*. Therefore when* $j = n$*, the minimization, in exact arithmetic, is over all* $\mathbb{R}^n$ *and thus* $\mathbf{x}_n$ *is the solution of* $\mathbf{A}\mathbf{x} = \mathbf{b}$*.*

Proof. — (Golub and Van Loan, 1996, p.490) ∎

The matrix of the system (6.18) has a tridiagonal form,

$$\mathbf{Q}_j^T \mathbf{A} \mathbf{Q}_j = \mathbf{T}_j = \begin{pmatrix} \delta_0 & \gamma_1 & & & \\ \gamma_1 & \delta_1 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \delta_{j-2} & \gamma_{j-1} \\ & & & \gamma_{j-1} & \delta_{j-1} \end{pmatrix}, \tag{6.19}$$

and the system can be solved cheaply using a factored decomposition (for instance using the `DPTSV` routine from the LAPACK library). An effective stopping criteria in finite precision arithmetic is to stop the algorithm when γ_j is reduced to a user-specified tolerance.

As for the conjugate gradient method, a split preconditioned Lanczos method can be derived, leading to Algorithm 6.5. The solution of the linear system is

Algorithm 6.5 Preconditioned Lanczos algorithm

$j = 0$
Given $\mathbf{r}_0$
$\mathbf{y}_0 = \mathbf{S}^T \mathbf{r}_0$
$\mathbf{q}_{-1} = 0$
for $j = 0, 1, 2, \ldots$ **do**
  $\gamma_j = ||\mathbf{y}_0||_2$
  $\mathbf{q}_j = \mathbf{y}_j / \gamma_j$
  $\delta_j = \mathbf{q}_j^T \mathbf{S}^T \mathbf{A} \mathbf{S} \mathbf{q}_j$
  $\mathbf{y}_{j+1} = \mathbf{S}^T \mathbf{A} \mathbf{S} \mathbf{q}_j - \delta_j \mathbf{q}_j - \gamma_j \mathbf{q}_{j-1}$
  $j = j + 1$
end for

then given by

$$\mathbf{x}_j = \mathbf{x}_0 + \mathbf{S} \mathbf{Q}_j \mathbf{w}_j, \tag{6.20}$$

where $\mathbf{w}_j$ is the solution of (6.18). It is also possible to derive a preconditioned Lanczos algorithm for a general preconditioner $\mathbf{P}$ (see Conn et al., 2000, p.104).

As said previously, the Lanczos algorithm can also be used to find eigenvalues and eigenvectors of a symmetric matrix. We have shown in equation (6.19) that the orthonormal basis $\mathbf{Q}_j$ can be used to transform the initial matrix $\mathbf{A}$ in a tridiagonal one $\mathbf{T}_j$. The idea of the Lanczos algorithm is to estimate the eigenpairs of $\mathbf{A}$ from the eigenpairs of $\mathbf{T}_j$. To this aim, the following eigenproblem of size j is solved

$$\mathbf{T}_j \mathbf{y} = \theta \mathbf{y},$$

and the j eigenpairs $(\theta_i, \mathbf{y}_i)$, $i = 1, \ldots, j$, are determined. They are called the *primitive Ritz pairs*. Their computation is cheap since the matrix $\mathbf{T}_j$ is

tridiagonal (the `DSTEQR` routine from LAPACK can be used). From these pairs, the *Ritz vectors* can be computed as $\mathbf{z}_i = \mathbf{Q}_j \mathbf{y}_i$, $i = 1, \ldots, j$, and the *Ritz pair* $(\theta_i, \mathbf{z}_i)$ can be constructed. Each Ritz pair can be related to an eigenpair $(\lambda_k, \mathbf{v}_k)$ of $\mathbf{A}$ such as

$$|\lambda_k - \theta_i| \leqslant ||\mathbf{A}\mathbf{z}_i - \theta_i \mathbf{z}_i||_2 = \gamma_j |\mathbf{e}_j^T \mathbf{y}_i|,$$

where $\mathbf{e}_j$ is the j-th canonical vector (see, Conn et al., 2000, p.101). Since the Ritz vectors $\mathbf{y}_i$ are normalized, we have $|\mathbf{e}_j^T \mathbf{y}_i| \leqslant 1$ and this bound shows that a small γ_j means that the Ritz pairs $(\theta_i, \mathbf{z}_i)$ are close to eigenvalues and eigenvectors of $\mathbf{A}$. However, a Ritz pair may also be close to an eigenpair of $\mathbf{A}$ if the j-th component of $\mathbf{y}_i$ is small. After n iterations, we have that $\mathbf{T}_n$ is similar to $\mathbf{A}$ and thus they share the same eigenvalues and eigenvectors.

There are strong relationships between the CG and the Lanczos algorithms. From Theorems 6.1 and 6.4, one can see that both methods minimize the quadratic function over the same sequence of nested Krylov subspaces $\mathcal{K}(\mathbf{r}_0, j)$. The CG and Lanczos algorithms are then equivalent, in exact arithmetic, for solving symmetric positive definite linear systems. Moreover, it is possible to determine the tridiagonal matrix $\mathbf{T}_j$ from the coefficients α_j and β_j computed in the CG algorithm. It is also possible to determine a $\mathbf{A}$-conjugate basis from the Krylov basis $\mathbf{Q}_j$ computed by the Lanczos method (see Conn et al., 2000, p.95). Due to their strong relationships, the term "*Conjugate Gradient-like*" (CG-like) method may refer simultaneously to the CG or to the Lanczos method.

6.3 Computational cost and diagnostics

In Algorithm 6.1, the Gauss-Newton method has been presented for solving the 4D-Var problem. This method states a sequence of symmetric positive definite linear systems

$$\mathbf{A}_k \delta\mathbf{x} = \mathbf{b}_k,$$

where $\mathbf{A}_k$ and $\mathbf{b}_k$ are given by (6.12) and (6.13). They can be solve by the CG method or the Lanczos method presented in Subsection 6.2.1 and 6.2.2. The computational cost of such methods is dominated by the matrix-vector products performed at each iteration (see line 11 of Algorithm 6.2 for the CG method and line 8 of Algorithm 6.4 for the Lanczos method). For any vector $\mathbf{p}$, the matrix-vector product is given by

$$\mathbf{A}_k \mathbf{p} = \mathbf{B}^{-1}\mathbf{p} + \mathbf{G}_k^T \mathbf{R}^{-1} \mathbf{G}_k \mathbf{p}. \tag{6.21}$$

While the first term can be computed straightforward, the second term must be computed with caution to avoid any extra computational cost. It can be

rewritten in detail as

$$\mathbf{G}_k^T \mathbf{R}^{-1} \mathbf{G}_k \mathbf{p} = \frac{1}{2} \sum_{i=1}^{N} \mathbf{M}_{i,0}^T \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{H}_i \mathbf{M}_{i,0} \mathbf{p}, \tag{6.22}$$

remembering the definition of $\mathbf{G}_k$ given in (3.9) and the block diagonal structure of $\mathbf{R}$ which contains each $\mathbf{R}_i$, for $i = 1, \ldots, N$. Note that the superscript k has been omitted for lighter notations. The Jacobian $\mathbf{M}_{i,0}$ of the nonlinear operator $\mathcal{M}_{i,0}$ and its transpose $\mathbf{M}_{i,0}^T$ are called the *tangent linear model* and the *adjoint model*, respectively. The computation of (6.22) is performed in two stages with the aim of minimizing the number of products by $\mathbf{M}_{i,0}$ and by $\mathbf{M}_{i,0}^T$ since these products dominate the computational cost. The first stage performs the integration of the tangent linear model from t_0 to t_N with the vector $\mathbf{p}$ as initial condition and stores the vectors

$$\mathbf{d}_i = \mathbf{R}_i^{-1} \mathbf{H}_i \mathbf{M}_{i,0} \mathbf{p},$$

for $i = 1, \ldots, N$. With this result, the sum (6.22) can be rewritten and expended as

$$\begin{aligned} \frac{1}{2} \sum_{i=1}^{N} \mathbf{M}_{i,0}^T \mathbf{H}_i^T \mathbf{d}_i &= \mathbf{M}_{1,0}^T \mathbf{H}_1^T \mathbf{d}_1 + \mathbf{M}_{2,0}^T \mathbf{H}_2^T \mathbf{d}_2 + \ldots + \mathbf{M}_{N,0}^T \mathbf{H}_N^T \mathbf{d}_N \\ &= \mathbf{M}_{1,0}^T \mathbf{H}_1^T \mathbf{d}_1 \\ &\quad + \mathbf{M}_{1,0}^T (\mathbf{M}_{2,1}^T \mathbf{H}_2^T \mathbf{d}_2 + \mathbf{M}_{2,1}^T (\ldots + \mathbf{M}_{N,N-1}^T \mathbf{H}_N^T \mathbf{d}_N) \ldots), \end{aligned}$$

using $\mathbf{M}_{i,0} = \prod_{j=0}^{j=i-1} \mathbf{M}_{j+1,j}$. The second stage of the computation then evaluates this last expression using an adjoint vector $\tilde{\mathbf{d}}$ and the following algorithm

1: $\tilde{\mathbf{d}} = 0$
2: **for** $i = N$ **to** 1 **do**
3: $\quad \tilde{\mathbf{d}} := \mathbf{M}_{i,i-1}^T \left(\tilde{\mathbf{d}} + \mathbf{H}_i^T \mathbf{d}_i \right)$
4: **end for**

This algorithm thus requires one suitably modified integration of the adjoint model from time t_N to t_0. At the end of this algorithm, the value of (6.22) is contained in the vector $\tilde{\mathbf{d}}$. In conclusion, the computation of the matrix vector product (6.21) requires one integration of the tangent linear model and of the adjoint model along the assimilation window.

It is common that no routine implements the matrix-vector product (6.21) in data assimilation software. However, there is often a routine which codes the evaluation of the incremental 4D-Var function value (6.16) with the evaluation of its gradient. In that case, the matrix vector product can be computed as

$$\begin{aligned} \mathbf{A}_k \mathbf{p} &= (\mathbf{A}_k \mathbf{p} - \mathbf{b}_k) - (-\mathbf{b}_k) \\ &= \nabla J^{\texttt{4D-inc}}(\mathbf{p}) - \nabla J^{\texttt{4D-inc}}(\mathbf{0}) \end{aligned}$$

where $\mathbf{0}$ represents a zero vector of size n. This simple trick is deduced using the derivative formula (A.7) with the definition (6.12) and (6.13).

During the solution of the linear system (the minimization of the incremental function), some user-oriented diagnostics can be performed. It is possible to compute cheaply, for each iterate $\mathbf{x}_j$, the incremental 4D-Var function value with its gradient. Indeed, the function value of the incremental 4D-Var at $\mathbf{x}_j$ can be formulated as

$$J^{\texttt{4D-inc}}(\mathbf{x}_j) = J^{\texttt{4D-inc}}(\mathbf{x}_0) + \frac{1}{2}(\mathbf{x}_j - \mathbf{x}_0)^T \mathbf{r}_0 \tag{6.23}$$

where $\mathbf{x}_0$ and $\mathbf{r}_0$ are defined in Algorithm 6.2 for the CG method and in Algorithm 6.4 with Theorem 6.4 for the Lanczos method (see Tshimanga, 2007, p.154, for further details). The function value at an iterate $\mathbf{x}_j$ is a useful diagnostic since it is related to the distance with the optimal solution $\mathbf{x}^*$ of the incremental problem as

$$J^{\texttt{4D-inc}}(\mathbf{x}_j) = \frac{1}{2}||\mathbf{x}_j - \mathbf{x}^*||^2_{\mathbf{A}_k} + J^{\texttt{4D-inc}}(\mathbf{x}^*),$$

where $\mathbf{A}_k$ is the matrix of the system (see Nocedal and Wright, 2006, p.113, for further explanations about this equality). The gradient of the incremental function at an iterate $\mathbf{x}_j$ is given by $\mathbf{A}_k\mathbf{x}_j - \mathbf{b}_k$. For the CG method, this information is directly computed in $\mathbf{r}_j$ at line 13 of Algorithm 6.2. For its part, the Lanczos method does not compute directly the gradient but it can be retrieved using the formula

$$\nabla J^{\texttt{4D-inc}}(\mathbf{x}_j) = (\mathbf{e}_j^T \mathbf{w}_j)\gamma_j \mathbf{q}_k,$$

where $\mathbf{e}_j$ is the jth canonical vector and where the other quantities are computed in Algorithm 6.4 and Theorem 6.4 (see Conn et al., 2000, p.100, for the derivation of the formula). However, when solving linear systems, the gradient norm tends to exhibit an erratic behavior and is not so useful to diagnostic the performance of the method (Nocedal et al., 2002).

6.4 Enhanced Gauss-Newton method using EOFs

The aim of this section is to develop an enhanced Gauss-Newton method by improving the speed of convergence of the iterative method used for the solution of the linear systems. It is well known (see Nocedal and Wright, 2006, for instance) that the rate of convergence of conjugate gradient-like methods is influenced by the starting point, the right-hand side and the actual distribution of the eigenvalues (e.g., the existence of eigenvalue clusters). Therefore, the development of good starting points and preconditioners is essential to obtain a better convergence rate.

Robert et al. (2005) and Robert et al. (2006) make use of the EOFs, introduced in Section 4.3.3, to define an appropriate reduced control space and an estimation of the background error covariance matrix. In their approach, an optimal solution is sought in the full space but the problem of the computational cost is addressed by considering a reduced 4D-Var to provide a relevant initial guess for the full space minimization. Their approach consists in performing a few number of iterations to approximately solve the *first* incremental 4D-Var (6.16) reduced to the subspace $\mathbf{L}_0$ containing the first r EOFs and with the sample covariance matrix $\mathbf{S}$, defined in (4.28), as background error covariance matrix approximation, that is

$$\min_{\delta\underline{\mathbf{x}}\in\mathbb{R}^r} \frac{1}{2}||\mathbf{L_0}\delta\underline{\mathbf{x}} - \mathbf{d}^{b0}||^2_{\mathbf{S}^{-1}} + \frac{1}{2}||\mathbf{G}_0\mathbf{L}_0\delta\underline{\mathbf{x}} - \mathbf{d}^{o0}||^2_{\mathbf{R}^{-1}}. \tag{6.24}$$

The prolongation of the approximate solution is then used as a starting point of a Gauss-Newton method to solve the full 4D-Var (3.6) where the background error covariance matrix $\mathbf{B}$ is used as preconditioner. They apply this reduced-order approach in the OPA model with its TDH configuration and the variational data assimilation package OPAVAR (Weaver et al., 2003). Note that Theorem 5.3 of Section 5.2 gives theoretical foundations for this approach. Indeed, under the assumptions of Theorem 5.3 and taking $\mathbf{d}^{b0} = \mathbf{x}^b$ and $\mathbf{d}^{o0} = \mathbf{y}$, the reduced 4D-Var (6.24) and the SEEK filter are equivalent (see the last comment at the end of Chapter 5 for more details). This motivates the relevance of approximately solving (6.24) to obtain an initial guess.

In what follows, we propose, and combine, two ways of exploiting the relevant information contained in the EOFs to accelerate the Gauss-Newton method. Firstly, to define an appropriate starting point for the CG-like method for the first incremental problem (6.16),

$$\min_{\delta\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}(\delta\mathbf{x}-\mathbf{d}^{b0})^T\mathbf{B}^{-1}(\delta\mathbf{x}-\mathbf{d}^{b0}) + \frac{1}{2}(\mathbf{G}_0\delta\mathbf{x}-\mathbf{d}^{o0})^T\mathbf{R}^{-1}(\mathbf{G}_0\delta\mathbf{x}-\mathbf{d}^{o0}), \tag{6.25}$$

we consider its reduced form based on the information contained in the r first EOFs stored in $\mathbf{L}_0 \in \mathbb{R}^{n\times r}$,

$$\begin{aligned}\min_{\delta\underline{\mathbf{x}}\in\mathbb{R}^r} = \frac{1}{2}&(\mathbf{L}_0\delta\underline{\mathbf{x}} - \mathbf{d}^{b0})^T\mathbf{B}^{-1}(\mathbf{L}_0\delta\underline{\mathbf{x}} - \mathbf{d}^{b0}) \\ &+ \frac{1}{2}(\mathbf{G}_0\mathbf{L}_0\delta\underline{\mathbf{x}} - \mathbf{d}^{o0})^T\mathbf{R}^{-1}(\mathbf{G}_0\mathbf{L}_0\delta\underline{\mathbf{x}} - \mathbf{d}^{o0}).\end{aligned} \tag{6.26}$$

This reduced incremental problem has been presented in Section 3.4 and differs from (6.24) only by the use of the background error covariance matrix $\mathbf{B}$ instead of the sample covariance matrix $\mathbf{S}$. It leads to the symmetric positive definite linear system

$$\mathbf{L}_0^T\mathbf{A}_0\mathbf{L}_0\delta\underline{\mathbf{x}} = \mathbf{L}_0^T\mathbf{b}_0, \tag{6.27}$$

using the notation from (6.12) and (6.13). The solution of (6.27), denoted by $\delta\underline{\mathbf{x}}_0$, is then prolongated in the full space, yielding $\mathbf{L}_0\delta\underline{\mathbf{x}}_0$. Observe that the prolongation of the solution $\delta\underline{\mathbf{x}}_0$ of (6.27) satisfies

$$\begin{aligned}\mathbf{L}_0\delta\underline{\mathbf{x}}_0 &= \mathbf{L}_0(\mathbf{L}_0^T\mathbf{A}_0\mathbf{L}_0)^{-1}\mathbf{L}_0^T\mathbf{b}_0 \\ &= \mathbf{L}_0(\mathbf{L}_0^T\mathbf{A}_0\mathbf{L}_0)^{-1}\mathbf{L}_0^T\mathbf{A}_0\delta\mathbf{x}_0,\end{aligned} \tag{6.28}$$

and corresponds to the solution of the full linear least-squares problem (6.25) projected onto $\mathbf{L}_0$ and orthogonal to $\mathbf{A}_0\mathbf{L}_0$ (see Saad, 2008, for a review on projections). In our approach, instead of performing only a few iterations on the reduced problem as in Robert et al. (2005), we compute (6.28) and choose it as a starting point to solve the full linear least-squares problem (6.25). This starting point is called the *Ritz-Galerkin starting point with respect to the subspace* $\mathbf{L}_0$ (van der Vorst, 2003). The matrix inverse in (6.28) is computed using a direct method since its dimension is equal to r and is very small in practice.

Considering again the first outer iteration of the Gauss-Newton method, the convergence rate of the CG-like method is strongly influenced by the distribution of the eigenvalues of $\mathbf{A}_0$ and is roughly bounded by a function of the condition number of $\mathbf{A}_0$ (see Theorems 6.2 and 6.3). Gratton et al. (2011) introduced a class of preconditioners called the Limited-Memory Preconditioner (LMP) which is related to the BFGS Hessian updating formula presented in equation (6.2). This class of preconditioners is based on the fact that when the objective function is quadratic,

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}^T - \mathbf{b}^T\mathbf{x},$$

the iterates generated by a line search method with the BFGS formula and optimal step lengths are identical to the iterates generated by the CG-like method (Nazareth, 1979). Assuming that r iterations have been performed, the correction pair $(\mathbf{z}_r, \mathbf{y}_r)$ involved in the BFGS formula is given by

$$\mathbf{z}_r = \mathbf{x}_r - \mathbf{x}_{r-1} = \alpha_r\mathbf{p}_r,$$

from line 12 of Algorithm 6.2, and by

$$\mathbf{y}_r = \mathbf{A}\mathbf{x}_r - \mathbf{b} - (\mathbf{A}\mathbf{x}_{r-1} - \mathbf{b}) = \mathbf{A}\mathbf{s}_r,$$

from the definition of $\mathbf{z}_r$. Using the $\mathbf{A}$-conjugate property of the $\mathbf{z}_i$ vectors, $i = 1, \ldots, r$, the BFGS formula (6.2) is given by

$$\mathbf{H}_r = \left(\mathbf{I}_n - \sum_{i=1}^{r}\frac{\mathbf{z}_i\mathbf{z}_i^T}{\mathbf{z}_i^T\mathbf{A}\mathbf{z}_i}\mathbf{A}\right)\mathbf{H}_0\left(\mathbf{I}_n - \sum_{i=1}^{r}\mathbf{A}\frac{\mathbf{z}_i\mathbf{z}_i^T}{\mathbf{z}_i^T\mathbf{A}\mathbf{z}_i}\right) + \sum_{i=1}^{r}\frac{\mathbf{z}_i\mathbf{z}_i^T}{\mathbf{z}_i^T\mathbf{A}\mathbf{z}_i}, \tag{6.29}$$

using the derivation from Gratton et al. (2011). If we now define an $n \times r$ matrix $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_r]$, one can observe that

$$(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1} = \operatorname{diag}\left((\mathbf{z}_i^T\mathbf{A}\mathbf{z}_i)^{-1}\right).$$

Relaxing the $\mathbf{A}$-conjugate property of $\mathbf{z}_i$, we obtain the general formulation of the LMP,

$$\mathbf{H}_r = [\mathbf{I}_n - \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{A}]\mathbf{H}_0 \\ [\mathbf{I}_n - \mathbf{A}\mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}\mathbf{Z}^T] + \mathbf{Z}(\mathbf{Z}^T\mathbf{A}\mathbf{Z})^{-1}\mathbf{Z}^T, \quad (6.30)$$

where the column of the matrix $\mathbf{Z} \in \mathbb{R}^{n\times r}$ are now assumed to contain linearly independent vectors. The initial inverse Hessian approximation $\mathbf{H}_0$ plays the role of a first-level preconditioner. The LMP presented in (6.30) is an inverse type preconditioner since it aims to approximate $\mathbf{A}^{-1}$. Its split preconditioner $\mathbf{H}_r = \mathbf{S}_r\mathbf{S}_r^T$ is given by

$$\mathbf{S}_r = \mathbf{I}_n - \mathbf{Z}\mathbf{R}^{-T}\mathbf{R}^{-1}\mathbf{Z}^T\mathbf{A} + \mathbf{Z}\mathbf{R}^{-T}\mathbf{U}^{-1}\mathbf{Z}^T,$$

where $\mathbf{R}$ and $\mathbf{U}$ are the lower triangular matrices coming, respectively, from the Cholesky decomposition $\mathbf{Z}^T\mathbf{A}\mathbf{Z} = \mathbf{R}\mathbf{R}^T$ and $\mathbf{Z}^T\mathbf{Z} = \mathbf{U}\mathbf{U}^T$. It is also possible to derive an equivalent forward preconditioner with its related split preconditioners (Tshimanga, 2007, p.50).

The first-level preconditioner $\mathbf{H}_0$ usually depends on the physics of the application. In variational data assimilation problems, it is equal to the background error covariance matrix $\mathbf{B}$ and is able to cluster most eigenvalues at 1. In order to improve the efficiency of this first-level preconditioner, the columns of the matrix $\mathbf{Z}$ in the LMP are built to contain directions in a low dimensional subspace that are left out by the first-level preconditioner and which slow down the convergence of the CG-like method. Recently, Tshimanga et al. (2008), used the LMP technique above in a Gauss-Newton context for solving the 4D-Var problem. Their idea is to exploit information gained when solving one system to build an LMP for the next system of the sequence. The information could be, for example, Ritz pairs or descent directions generated by the CG-like method. Their method however is unable to exploit the LMP technique to design a preconditioner for the first system to solve in the sequence, since in this case no information is already available through a CG-like method. We propose here such a preconditioner, based on EOFs. More precisely, we choose as preconditioner for the first system the LMP

$$\mathbf{H}_r = [\mathbf{I}_n - \mathbf{L}_0(\mathbf{L}_0^T\mathbf{A}_0\mathbf{L}_0)^{-1}\mathbf{L}_0^T\mathbf{A}_0]\mathbf{B} \\ [\mathbf{I}_n - \mathbf{A}_0\mathbf{L}_0(\mathbf{L}_0^T\mathbf{A}_0\mathbf{L}_0)^{-1}\mathbf{L}_0^T] + \mathbf{L}_0(\mathbf{L}_0^T\mathbf{A}_0\mathbf{L}_0)^{-1}\mathbf{L}_0^T, \quad (6.31)$$

where $\mathbf{L}_0 \in \mathbb{R}^{n\times r}$ contains the first r EOFs. The cost of building this preconditioner is dominated by the product $\mathbf{A}_0\mathbf{L}_0$ and is thus available without any extra computational cost since this product has been computed for the starting point (6.28). In the numerical experiments that follow, we assume that the Hessian approximations $\mathbf{A}_k$ do not change significantly from one outer iteration to the next and keep the preconditioner unchanged for each system, since it is available from the first to the last system of the sequence.

6.5 Numerical experiments

In what follows, we illustrate the numerical behavior of our enhanced Gauss-Newton method which uses the appropriate starting point and the limited memory preconditioner, both based on EOFs information, presented in the previous section. The model we consider is the one-dimensional shallow water system describing the flow of a fluid over an obstacle (see Lawless et al. (2003)). The governing equation can be written as

$$\begin{cases} \dfrac{Du}{Dt} + \dfrac{\partial \phi}{\partial z} &= -\mathrm{g}\, \dfrac{\partial \bar{h}}{\partial z}, \\ \dfrac{D\phi}{Dt} + \phi \dfrac{\partial u}{\partial z} &= 0, \end{cases}$$

where $\frac{D}{Dt} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial z}$ is the material derivative and g is the gravitational constant. In these equations, $u = u(z,t)$ is the velocity of the fluid, $\phi = \mathrm{g}h(z,t)$ is the geopotential, with $h(z,t) > 0$, the depth of the fluid above the orography, and $\bar{h} = \bar{h}(z)$ is the height of the bottom orography. The problem is defined on the domain $z \in [0, \ell]$, with the periodic boundary condition $z(0) = z(\ell)$, and for $t \in [0, T]$. The height of the obstacle is given by

$$\bar{h}(z) = \begin{cases} \bar{h}_c \left(1 - \frac{(z - \frac{\ell}{2})^2}{a^2}\right) & \text{for } \ 0 \leqslant |z - \frac{\ell}{2}| \leqslant a, \\ 0 & \text{otherwise,} \end{cases}$$

where $\bar{h}_c$ is the maximum height of the obstacle and a is half the length over which the base of the obstacle extends. In our experiments, a is set to $0.4m$ and $\bar{h}_c$ to $0.05m$. The values of u and ϕ are specified everywhere at the initial time as

$$u(z,0) = u_0(z) = u_0,$$

$$\phi(z,0) = \phi_0(z) = \mathrm{g}\ (h_0 - \bar{h}(z)),$$

with $u_0 = 0.1m/s$ and $h_0 = 0.2m$. At time $t = 0$, the fluid is impulsively put in motion with the constant velocity u_0. From this impulse, a wave motion develops and moves away from the obstacle in both directions. In our example, the domain is defined to be periodic over 250 grid points, with a distance of $\Delta z = 0.01m$ between them, so that $z \in [0m, 2.5m]$. For simplicity, we set the gravitational constant g to $10m/s^2$. The time-step Δt for the model integration is $4.6 \times 10^{-3} s$. The state vector of the system at a time t is given by the velocity and the geopotential at each of the 250 grid points and is thus of dimension $n = 500$.

The framework for the numerical tests is the classical twin experiment. The system period, i.e., the average time for a wave to cover the whole basin, is empirically estimated to 400 time-steps. It is divided into 10 assimilation

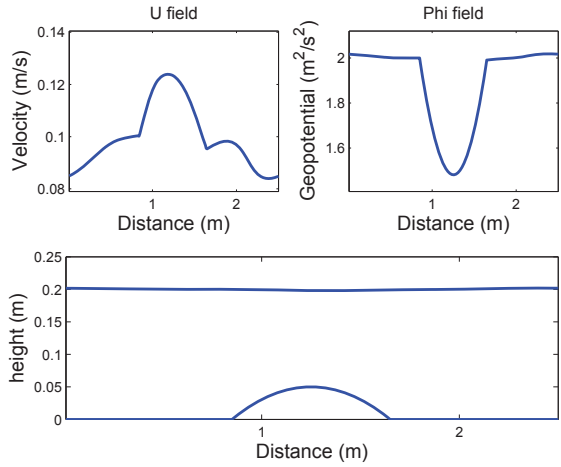

Figure 6.1 — *Velocity, geopotential and water level at a given time for the shallow water model.*

windows and we solve the data assimilation problem for the first assimilation window only, hence composed of 40 time-steps. We first compute reference states by integrating the model from an arbitrary initial condition during 40 time-steps. Observations are then generated from the reference states by adding Gaussian noises. To build the sample covariance matrix $\mathbf{S}$ given by (4.28) and required to get $\mathbf{L}_0$, one computes a set of $l = 50$ state vectors by integrating the model during 400 time-steps from an arbitrary initial condition occurring before the assimilation window, storing the state vectors produced every 8th time-step. Note that the background $\mathbf{x}^b$ is computed as the average of these 50 state vectors, while the background covariance matrix $\mathbf{B}$ is computed using a Laplacian-based correlation model, with a correlation lengthscale equal to $1.5m$ and a variance of 1.0×10^{-3}. The spectral decomposition (4.29) of $\mathbf{S}$ is then applied to compute the EOFs basis. The percentage of variation accounted for by the first r EOFs, defined in (4.31), is plotted in Figure 6.2 (for $r = 1, \ldots, 50$). One can see that retaining the first five EOFs enables to explain 80% of the system's variability. This choice for r is a good compromise since it allows to account for most of the variation without increasing too much the dimension of the subspace spanned by the selected EOFs. Note that all these parameters are chosen according to the choices made for operational data assimilation problems.

The set of experiments is designed to illustrate the impact of the Ritz-Galerkin starting point (6.28) and of the LMP (6.31) when the subspace spanned by $\mathbf{L_0}$ is defined by the five dominant EOFs. We allow for three outer Gauss-Newton iterations with five inner CG-like iterations to solve each linear system,

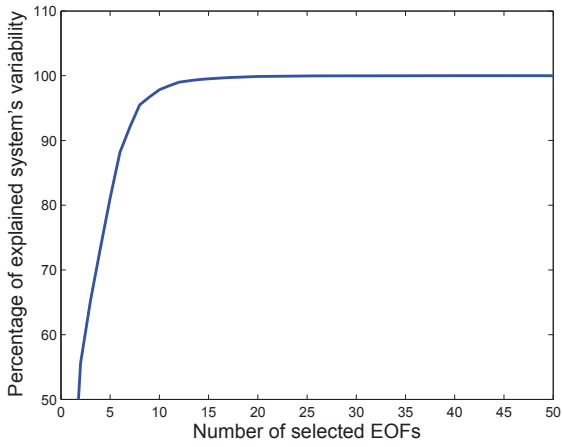

Figure 6.2 — *Percentage of explained system's variability versus the number of selected EOFs.*

which corresponds to realistic heuristics for operational 4D-Var problems. We consider a first set of three experiments with an increasing number of observations (see Figures 6.3 to 6.5). These observations are generated at each time-step, implying that $N = 40$. A velocity and geopotential observation is generated at the 50th grid point for the first experiment ($p_i = 2$), at the 50th and 200th grid points for the second experiment ($p_i = 4$), and at the 50th, 125th and 200th grid points for the third one ($p_i = 6$). In these experiments, the observation error covariance matrix $\mathbf{R}$ is set to a multiple μ of the identity matrix, with $\mu = 10^{-4}$. These experiments yield an observation vector space of dimension $p = 80$, $p = 160$ and $p = 240$, respectively. We next consider a second set of two experiments in which μ is set to $\mu = 10^{-5}$ and $\mu = 10^{-3}$, successively, for the case $p = 240$ (see Figures 6.6 and 6.7).

Figures 6.3 to 6.7 show the history of the nonlinear and incremental cost functions for the three outer iterations (the nonlinear with circles and the incremental with lines). Note that the nonlinear function value is available only at outer iterations when a linearization is performed in the Gauss-Newton algorithm. For the incremental cost functions, the curves are placed one after the other in sequence and the inner iterations are cumulated. The dash-dot curve is obtained using a basic Gauss-Newton algorithm with the background covariance matrix $\mathbf{B}$ as preconditioner in the CG-like method. The dotted curve and the dashed curve are obtained using the same approach, but with the Ritz-Galerkin starting point (6.28) for the first system. No second-level preconditioner has been added to generate the dotted curve, while the LMP (6.31) has been used as second-level preconditioner to get the dashed curve.

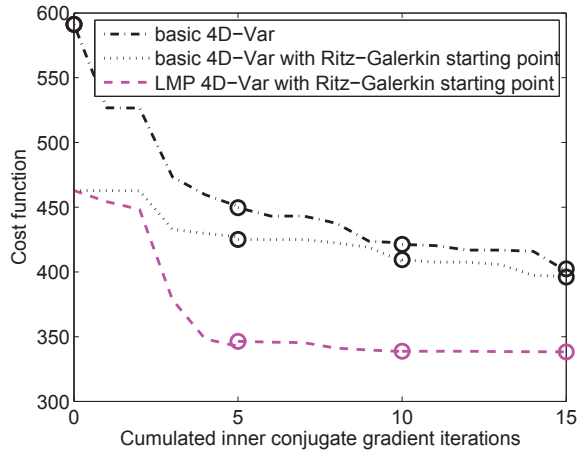

Figure 6.3 — *Convergence curves for* $p = 80$ *and* $\mu = 10^{-4}$.

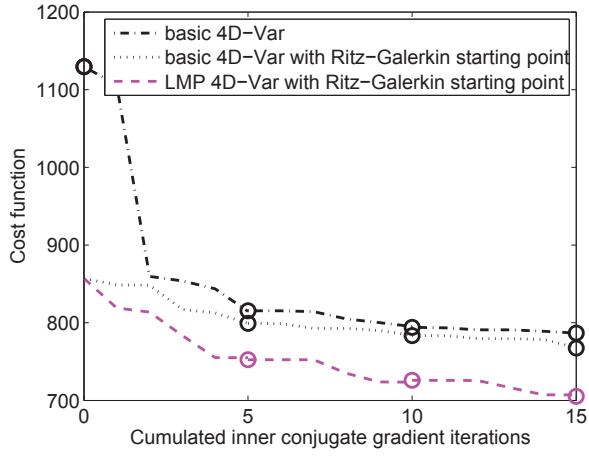

Figure 6.4 — *Convergence curves for* $p = 160$ *and* $\mu = 10^{-4}$.

A first look at these pictures show that the Ritz-Galerkin starting point provides a good reduction for the first quadratic problem and globally improves the convergence of the Gauss-Newton method. The second-level preconditioner however substantially improves the reduction in the nonlinear function. These remarks hold for all five experiments, showing the small influence of the number of observations and of their accuracy on the comparative behaviour of the three approaches. But one needs to be careful as the methods do not have the same cost for the whole process. Indeed, this cost is dominated by the number of

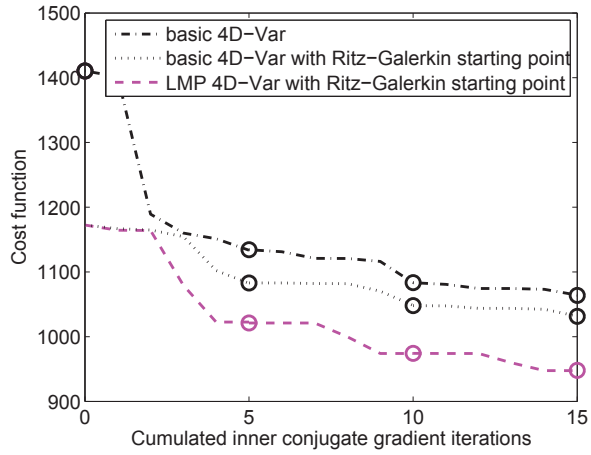

Figure 6.5 — *Convergence curves for* $p = 240$ *and* $\mu = 10^{-4}$.

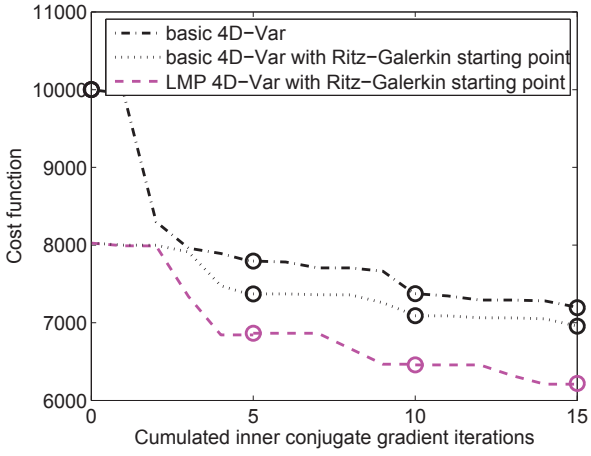

Figure 6.6 — *Convergence curves for* $p = 240$ *and* $\mu = 10^{-5}$.

matrix-vector products with the system's matrices $\mathbf{A}_k$ $(k = 1, 2, 3)$ defined in (6.21). Remembering that the computation of the Ritz-Galerkin starting point requires five (r) matrix-vector products to evaluate $\mathbf{A}_0\mathbf{L}_0$ and that the LMP (6.31) is available without any extra cost, it makes sense to compare the results obtained after 15 cumulated inner iterations of the basic approach with that obtained after 10 cumulated inner iterations of the two others. Table 6.1 reports the nonlinear function values extracted from Figures 6.3 to 6.7 after a computational cost of 15 matrix-vector products for the three methods (denoted

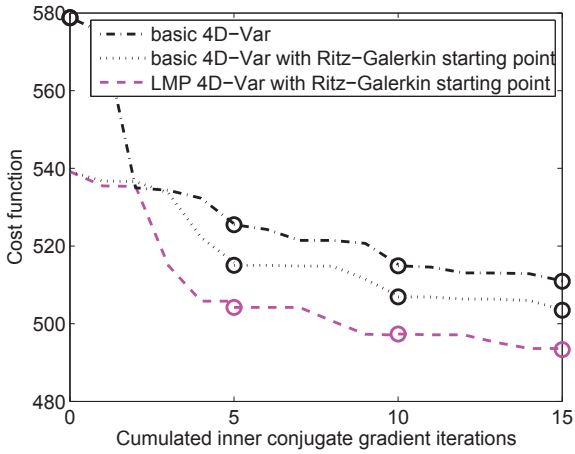

Figure 6.7 — *Convergence curves for* $p = 240$ *and* $\mu = 10^{-3}$.

"Basic", "RG" and "RG+LMP", respectively, in Table 6.1). From this table,

Experiment	Basic	RG	RG+LMP
$p = 80$, $\mu = 10^{-4}$	402.3	409.4	338.8
$p = 160$, $\mu = 10^{-4}$	786.8	783.4	726.0
$p = 240$, $\mu = 10^{-4}$	1064.2	1048.8	964.1
$p = 240$, $\mu = 10^{-5}$	7194.3	7090.1	6458.7
$p = 240$, $\mu = 10^{-3}$	510.9	506.9	497.4

Table 6.1 — *Nonlinear function values after a same computational cost of 15 matrix-vector products for the three methods.*

we see that the use of the Ritz-Galerkin starting point improves the reduction obtained in the nonlinear function for the same computational effort (except for the first experiment). However, it clearly appears that adding a second-level preconditioning generates a more substantial improvement, showing the power of combining the Ritz-Galerkin starting point with an LMP of the form (6.31).

We can partly explain the good performance of adding the LMP by looking at the spectrum of $\mathbf{A}_0$, the matrix of the first system. We have plotted in Figure 6.8 the spectrum of $\mathbf{A}_0$ and of its two preconditioned versions, $\mathbf{B}\mathbf{A}_0$ and $\mathbf{H}_0\mathbf{A}_0$, for the case $p = 80$ and $\mu = 10^{-4}$. The curves for the two preconditioned versions being hardly distinguishable on Figure 6.8, we have also plotted their 20 largest eigenvalues in Figure 6.9, to make the comparison easier. The matrix $\mathbf{A}_0$ has a spectrum that does not exhibit any gap and a condition number of 9.70×10^7. By (6.12), one has that the preconditioned matrix $\mathbf{B}\mathbf{A}_0$ is given by $\mathbf{I}_n + \mathbf{B}\mathbf{G}_0^T\mathbf{R}^{-1}\mathbf{G}_0$, that is, the identity matrix augmented by a symmetric

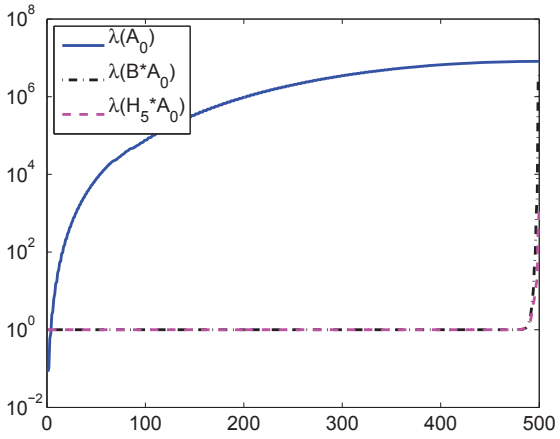

Figure 6.8 — *Spectrum of* $\mathbf{A}_0$, $\mathbf{BA}_0$ *and* $\mathbf{H}_5\mathbf{A}_0$.

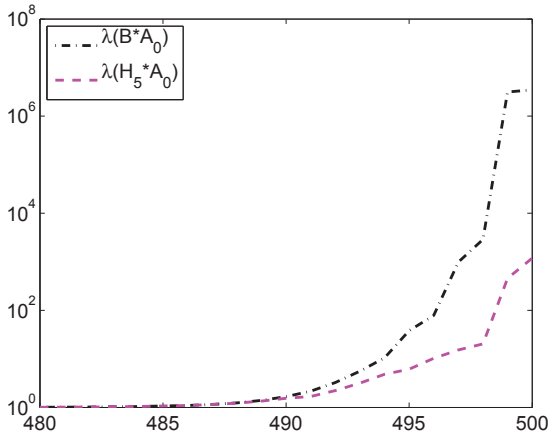

Figure 6.9 — *Twenty largest eigenvalues of* $\mathbf{BA}_0$ *and* $\mathbf{H}_5\mathbf{A}_0$.

positive semidefinite matrix. Its spectrum thus becomes bounded below by 1 and has a cluster of eigenvalues at 1 of size at least $\max\{0, n-p\}$, where p is the dimension of the observation vector $\mathbf{y}$. For our example, the size of this cluster is thus at least $\max\{0, 500-80\} = 420$, and the condition number is 3.48×10^6. The condition number of the preconditioned matrix $\mathbf{H}_0\mathbf{A}_0$ falls down to 1.19×10^3, while its cluster of eigenvalues at 1 keeps at least the same size than that of $\mathbf{BA}_0$ (Gratton et al., 2011). Since the convergence rate of the CG-like method depends, to a large extend, on the condition number and on

the clustering of the eigenvalues, this spectral analysis gives some insight on the impact the LMP (6.31) has on the first linear system.

We conclude the numerical experiments by illustrating the impact of the number of EOFs used to construct the Ritz-Galerkin starting point and the LMP. Figure 6.10 shows the nonlinear function values obtained after three

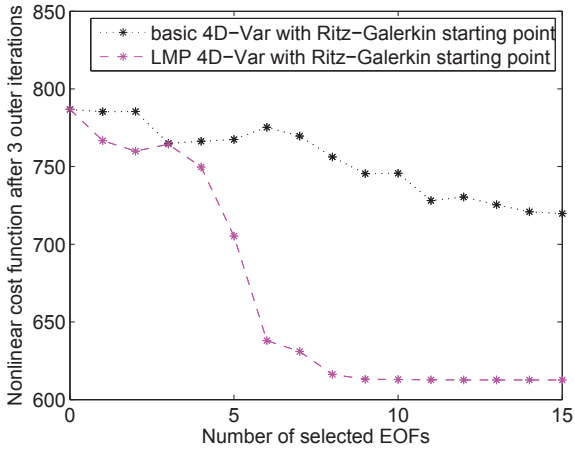

Figure 6.10 — *Nonlinear function values obtained after the whole process when using an increasing number of EOFs in the Ritz-Galerkin starting point and the LMP.*

outer iterations, i.e., 15 cumulated CG-like iterations, for the case $p = 80$ and $\mu = 10^{-4}$. The results for the approaches using a Ritz-Galerkin starting point without and with the LMP are reported for a number of selected EOFs varying from 0 to 15. It is important to note that in this picture, the overall cost in terms of matrix-vector products is not constant but increases with the number of EOFs. Figure 6.10 first illustrates the significant contribution of using a second-level preconditioning, independently of the number of EOFs used to build the Ritz-Galerkin starting point and the LMP. It also shows that the choice for $r = 5$ based on the percentage of explained system's variability is relevant. Indeed, using less than 4 EOFs leads to a poor improvement, while using more than 8 EOFs does not bring a significant additional improvement in the reduction of the objective function.

Chapter 7

Derivative-free approach for the 4D-Var

In the previous chapter, we have presented the Gauss-Newton method which uses the information contained in the Jacobian matrices to solve the nonlinear 4D-Var problem. Although the Kalman filter has been initially developed for linear models, it has been adapted to handle nonlinear ones. It leads up to the extended Kalman filter which is presented in the beginning of the chapter. Since this filter performs linearizations of the operators using Jacobian matrices, it is not suited for highly nonlinear dynamics. To address this problem, a class of ensemble-based Kalman filters proposes an alternative where the state of the system is estimated by a probabilistic density function computed from an ensemble of well-chosen state vectors. This class has the advantage to be a derivative-free alternative since it does not require the computation of Jacobian matrices. Few ensemble-based Kalman filters are described including the *Ensemble Kalman filter* (EnKF) which is particularly interesting since it is adapted for large systems.

To our knowledge, no derivative-free variant of the variational approach has been proposed so far. Yet the optimization community has been quite active in developing derivative-free optimization (DFO) methods designed to solve nonlinear problems arising from the engineering community. Maybe is this due to the fact that DFO algorithms are not yet able to handle problems larger than few tens of variables at once. The rest of the chapter is devoted to the presentation of an attempt to solve the 4D-Var problem using a derivative-free approach. It is based on the construction of a sequence of well-chosen and low dimensional subspaces and exploits reduction techniques from EOFs. An academical shallow water model is used to illustrate the behavior of our derivate-free approach and to compare it with a basic EnKF and with a classical Gauss-Newton method. Other numerical experiments using the NEMO

framework are presented in Chapter 8.

7.1 The extended Kalman filter

In Chapter 4, the derivation of the Kalman filter with its optimal statistical properties is clean and powerful. Nevertheless, most real life applications do not behave linearly and are modeled using nonlinear operators. Trying to estimate these models with the linear filter theory will not give good results and nonlinear filters are needed. This branch is more complex and less mature than linear filtering.

The *Extended Kalman Filter* (EKF) is a straightforward extension of the classical Kalman filter and solves the issue of nonlinear operators by linearizing them around well-chosen points. To present this approach, we recall from Chapter 2 that the model operator from time t_{i-1} to time t_i and the observation $\mathbf{y}_i$ are given, in the nonlinear case, by

$$\begin{cases} \mathbf{x}_i^t = \mathcal{M}_{i,i-1}\mathbf{x}_{i-1}^t \\ \mathbf{y}_i = \mathcal{H}_i\mathbf{x}_i^t + \boldsymbol{\epsilon}_i^o, \end{cases} \tag{7.1}$$

where the superscript t denotes the true state of the system. In the Kalman filter (Algorithm 4.1), one can remark that the forecast and the analysis are computed from $\mathbf{x}_{i-1}^a$ and $\mathbf{x}_i^f$, respectively. Thus, it seems natural to linearize the model operator around $\mathbf{x}_{i-1}^a$ and the observation operator around $\mathbf{x}_i^f$. The Taylor series expansion to the first order of the model operator $\mathcal{M}_{i,i-1}$ around $\mathbf{x}_{i-1}^a$ can be formulated as

$$\begin{aligned} \mathcal{M}_{i,i-1}\mathbf{x}_{i-1}^t &\simeq \mathcal{M}_{i,i-1}\mathbf{x}_{i-1}^a + \mathbf{M}_{i,i-1}\left(\mathbf{x}_{i-1}^t - \mathbf{x}_{i-1}^a\right) \\ &= \mathbf{M}_{i,i-1}\mathbf{x}_{i-1}^t + \left(\mathcal{M}_{i,i-1}\mathbf{x}_{i-1}^a - M_{i,i-1}\mathbf{x}_{i-1}^a\right) \\ &= \mathbf{M}_{i,i-1}\mathbf{x}_{i-1}^t + \mathbf{u}_{i-1}, \end{aligned} \tag{7.2}$$

where $\mathbf{u}_{i-1} = \mathcal{M}_{i,i-1}\mathbf{x}_{i-1}^a - M_{i,i-1}\mathbf{x}_{i-1}^a$ is a known vector. In the same way, the Taylor series expansion to the first order of the observation operator $\mathcal{H}_i$ around x_i^f is given by

$$\begin{aligned} \mathcal{H}_i\mathbf{x}_i^t &\simeq \mathcal{H}_i\mathbf{x}_i^f + \mathbf{H}_i\left(\mathbf{x}_i^t - \mathbf{x}_i^f\right) \\ &= \mathbf{H}_i\mathbf{x}_i^t + \left(\mathcal{H}_i\mathbf{x}_i^f - \mathbf{H}_i\mathbf{x}_i^f\right) \\ &= \mathbf{H}_i\mathbf{x}_i^t + \mathbf{z}_i, \end{aligned} \tag{7.3}$$

where $\mathbf{z}_i = \mathcal{H}_i\mathbf{x}_i^f - \mathbf{H}_i\mathbf{x}_i^f$ is a known vector too. Using both approximations (7.2) and (7.3) in the definition of the system (7.1), we obtain

$$\begin{cases} \mathbf{x}_i^t = \mathbf{M}_{i,i-1}\mathbf{x}_{i-1}^t + \mathbf{u}_i \\ \mathbf{y}_i = \mathbf{H}_i\mathbf{x}_i^t + \mathbf{z}_i + \boldsymbol{\epsilon}_i^o, \end{cases} \tag{7.4}$$

which is the linearization of (7.1) around $\mathbf{x}_{i-1}^a$ and x_i^f. Since this system is linear, the Kalman filter presented in Algorithm 4.1 can be applied taking care of the fact that two known vectors, $\mathbf{u}_i$ and $\mathbf{z}_i$, have appeared and cause some modifications. Indeed, at the i-th iteration of the Kalman filter, the analysis $\mathbf{x}_{i-1}^a$ is now propagated using the model transition defined in (7.4), which gives

$$\begin{aligned}\mathbf{x}_i^f &= \mathbf{M}_{i,i-1}\mathbf{x}_{i-1}^a + \mathbf{u}_i \\ &= \mathbf{M}_{i,i-1}\mathbf{x}_{i-1}^a + \mathcal{M}_{i,i-1}\mathbf{x}_{i-1}^a - \mathbf{M}_{i,i-1}\mathbf{x}_{i-1}^a \\ &= \mathcal{M}_{i,i-1}\mathbf{x}_{i-1}^a.\end{aligned}$$

For its part, the propagation of the analysis error covariance matrix can be derived as

$$\begin{aligned}\mathbf{P}_i^f &= E\left[(\mathbf{x}_i^f - \mathbf{x}_i^t)(\mathbf{x}_i^f - \mathbf{x}_i^t)^T\right] \\ &= E\Big[\left(\mathbf{M}_{i,i-1}\mathbf{x}_{i-1}^a + \mathbf{u}_i - \mathbf{M}_{i,i-1}\mathbf{x}_{i-1}^t - \mathbf{u}_i\right) \\ &\qquad\qquad \left(\mathbf{M}_{i,i-1}\mathbf{x}_{i-1}^a + \mathbf{u}_i - \mathbf{M}_{i,i-1}\mathbf{x}_{i-1}^t - \mathbf{u}_i\right)^T\Big] \\ &= E\Big[\mathbf{M}_{i,i-1}(\mathbf{x}_{i-1}^a - \mathbf{x}_{i-1}^t)(\mathbf{x}_{i-1}^a - \mathbf{x}_{i-1}^t)^T\mathbf{M}_{i,i-1}^T\Big] \\ &= \mathbf{M}_{i,i-1}E\Big[(\mathbf{x}_{i-1}^a - \mathbf{x}_{i-1}^t)(\mathbf{x}_{i-1}^a - \mathbf{x}_{i-1}^t)^T\Big]\mathbf{M}_{i,i-1}^T \\ &= \mathbf{M}_{i,i-1}\mathbf{P}_{i-1}^a\mathbf{M}_{i,i-1}^T.\end{aligned}$$

From the definition of the observation $\mathbf{y}_i$ in (7.4), the departure vector used in the correction step at line 9 of Algorithm 4.1 is now given by $\mathbf{y}_i - (\mathbf{H}_i\mathbf{x}_i^f + \mathbf{z}_i)$ and can be reformulated as

$$\begin{aligned}\mathbf{y}_i - (\mathbf{H}_i\mathbf{x}_i^f + \mathbf{z}_i) &= \mathbf{y}_i - \mathbf{H}_i\mathbf{x}_i^f - \mathcal{H}_i\mathbf{x}_i^f + \mathbf{H}_i\mathbf{x}_i^f \\ &= \mathbf{y}_i - \mathcal{H}_i\mathbf{x}_i^f.\end{aligned}$$

The other equations of the Kalman filter are not affected by the vectors $\mathbf{u}_i$ and $\mathbf{z}_i$. The extended Kalman filter is summarized in Algorithm 7.1. Nowadays, it is used as a standard in navigation systems (Strang and Borre, 1997) even if it does have a few flaws. Due to the approximations (7.2) and (7.3), the extended Kalman filter is not an optimal estimator unlike its linear counterpart. This filter is only reliable for systems that are almost linear with respect to the time discretization. If the model and observation operators are highly nonlinear, the filter may quickly diverge (Simon, 2006).

The *Iterated Kalman Filter* (IKF) (Bell and Cathey, 1993) is directly related to the extended Kalman filter and solves the problem of nonlinearities in the observation operators by performing multiple analysis steps. As in the extended Kalman filter (see Algorithm 7.1), it first performs a linearization of the observation operator around $\mathbf{x}_i^f$ since it is the current best estimate and

Algorithm 7.1 Extended Kalman filter with perfect nonlinear model operators

$\mathbf{x}_0^a = \mathbf{x}^b$
$\mathbf{P}_0^a = \mathbf{B}$
for $i = 1$ to N **do**
  (* **Forecast step** *)
  $\mathbf{x}_i^f = \mathcal{M}_{i,i-1}\mathbf{x}_{i-1}^a$
  $\mathbf{P}_i^f = \mathbf{M}_{i,i-1}\mathbf{P}_{i-1}^a\mathbf{M}_{i,i-1}^{\mathrm{T}}$
  (* **Analysis step** *)
  $\mathbf{K}_i = \mathbf{P}_i^f\mathbf{H}_i^{\mathrm{T}}(\mathbf{H}_i\mathbf{P}_i^f\mathbf{H}_i^{\mathrm{T}} + \mathbf{R}_i)^{-1}$
  $\mathbf{x}_i^a = \mathbf{x}_i^f + \mathbf{K}_i(\mathbf{y}_i - \mathcal{H}_i\mathbf{x}_i^f)$
  $\mathbf{P}_i^a = (\mathbf{I}_n - \mathbf{K}_i\mathbf{H}_i)\mathbf{P}_i^f$
end for

computes the analysis $\mathbf{x}_i^a$. Nevertheless, it does not go directly to the next forecast step but computes a new analysis step where the observation operator is linearized, this time, around $\mathbf{x}_i^a$, the new best estimate. This process can be repeated as many times as desired. For the majority of problems, most of the possible improvements is obtained by only performing one extra linearization. Unfortunately, the IKF cannot solve problems with high nonlinearities in the model operator since extra linearizations are not applicable for the forecast step.

For the sake of completeness, we present the SEEK filter (Algorithm 4.2) in the nonlinear case. It is summarized in Algorithm 7.2 and is derived from the extended Kalman filter using the same approach as in Subsection 4.3.1.

Algorithm 7.2 SEEK filter with perfect nonlinear model operators

$\mathbf{x}_0^a = \mathbf{x}^b$
$\mathbf{P}_0^a = \mathbf{L}_0\mathbf{U}_0\mathbf{L}_0^{\mathrm{T}}$
for $i = 1$ to N **do**
  (* **Forecast step** *)
  $\mathbf{x}_i^f = \mathcal{M}_{i,i-1}\mathbf{x}_{i-1}^a$
  $\mathbf{L}_i = \mathbf{M}_{i,i-1}\mathbf{L}_{i-1}$
  $\mathbf{P}_i^f = \mathbf{L}_i\mathbf{U}_{i-1}\mathbf{L}_i^{\mathrm{T}}$
  (* **Analysis step** *)
  $\mathbf{U}_i = (\mathbf{U}_{i-1}^{-1} + \mathbf{L}_i^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}\mathbf{R}_i^{-1}\mathbf{H}_i\mathbf{L}_i)^{-1}$
  $\mathbf{K}_i = \mathbf{L}_i\mathbf{U}_i\mathbf{L}_i^{\mathrm{T}}\mathbf{H}_i^{\mathrm{T}}\mathbf{R}_i^{-1}$
  $\mathbf{x}_i^a = \mathbf{x}_i^f + \mathbf{K}_i(\mathbf{y}_i - \mathcal{H}_i\mathbf{x}_i^f)$
  $\mathbf{P}_i^a = \mathbf{L}_i\mathbf{U}_i\mathbf{L}_i^{\mathrm{T}}$
end for

7.2 Computing the Jacobian

In the previous section, we have presented the EKF which is based on the linearization of the model and observation operators and which involves the computation of their Jacobian matrices. We have shown, from the last chapter, that the variational approach also requires the computation of these Jacobian matrices to generate the sequence of incremental problems. The computation of such matrices is not trivial for operational ocean and atmosphere models. The main issues are highlighted in this section and their costs are analyzed. Actually, due to the very large size of the vectors in weather forecasting, the Jacobian matrices and their transposes are not stored explicitly but the construction of a *tangent code* and of an *adjoint code* are requested to compute Jacobian-vector products. We next shortly explain how these nontrivial codes are built. The development of a numerical oceanic or atmospheric model is usually done in three steps. The analytical differential equations are first formulated. Then a discretization scheme is chosen and the discrete difference equations are constructed. Finally an algorithm is implemented that solves the discrete equations using a programming language, producing a *direct code*. The tangent and adjoint codes may be theoretically implemented after any of these three steps (Giering and Kaminski, 1998). In practice however, oceanic or atmospheric models have been quite often conceived without taking the coding of the derivatives into account and the tangent and adjoint codes were, in that case, derived from the direct code. To do so, during the eighties and the nineties, hand coding based on the "chain rule" on the direct code was used. This task was extremely error prone and time consuming. Moreover, for any change in the direct code, both the tangent and adjoint codes had to be rewritten. For these reasons, such codes were rarely developed and only for simplified models. As an exception, the tangent and adjoint codes for the atmospheric model at ECMWF and for the ocean general circulation model OPA have been written and maintained by hand (Tber et al., 2007). Algorithmic, or automatic, differentiation (AD) techniques then appeared, allowing to accurately and efficiently evaluate derivatives for functions or models given as computer programs (Griewank and Walther, 2008). With automatic differentiation, any change in the direct code can be propagated to the tangent and adjoint codes at almost no additional cost. Recently, a number of AD software have been developed that are capable of generating reliable tangent and adjoint codes (TAF, TAMC, Tapenade, Odyssee). Unfortunately, these codes are often less efficient than hand written codes because they cannot exploit some tricks that could be made manually. This drawback seems to be damped with the latest AD tools (Giering and Kaminski, 1998). Today the tangent and adjoint codes need between 2 and 5 times the execution time of the direct code. This ratio decreases with the new advancements in automatic differentiation combined with a careful preparation of the direct code (Müller and Cusdin, 2005).

7.3 The ensemble-based Kalman filters

The *ensemble-based Kalman filters* are a broad class of nonlinear filters which are designed to handle nonlinear operators without relying on the computation of their Jacobian matrices. They are based on a *recursive Bayesian approach* where the state vector of the system is assumed to be a random vector which follows a given but unknown Probabilistic Density Function (PDF). The ensemble-based Kalman filters keep the Kalman filter philosophy unchanged but generate a sequence of PDFs from the information contained in an ensemble of state vectors which are propagated and corrected along the data assimilation window. Since the PDF contains all the available statistical information, it may be regarded as the complete solution of the estimation problem in comparison with the Kalman filter approach which computes only the first two statistical moments, namely the mean and the covariance matrix.

Depending on the assumptions made on the PDFs and on the ensemble of state vectors, different filters can be derived. We present three of them. The first one is called the *particle filter* and implements the Bayesian approach for any general PDFs. It randomly generates a given number of state vectors called particles and uses a sequential Monte-Carlo technique to estimate the PDF from the information contained in the particles (Ristic et al., 2004). However, this filter is not directly applicable since it requires a large number of particles to estimate entirely the PDF (Snyder et al., 2008).

For their part, the two other ensemble-based Kalman filters, namely the *unscented Kalman filter* and the *ensemble Kalman filter*, generate PDFs with the assumption that they follow multivariate normal distributions. The main advantage of such a distribution is that it is fully determined by its mean and covariance matrix. Thus, it simplifies the computation process since the entire PDF must not be estimated but only the mean and the covariance matrix. Before presenting the two methods, we show that this approach is equivalent to the Kalman filter if the model and observation operators are linear. This equivalence is well-known (see, for example, Anderson and Moore, 1979). The proof is based on a statistical result which calculates the conditional distribution between two vectors which follow multivariate normal distributions.

Theorem 7.1 *Let* $\mathbf{z} \sim N_{n+p}(\mathbf{m}, \Sigma)$ *be a multivariate normal random vector of mean* $\mathbf{m} \in \mathbb{R}^{n+p}$ *and of covariance matrix* $\Sigma \in \mathbb{R}^{(n+p)\times(n+p)}$ *partitioning as*

$$\mathbf{z} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}, \quad \mathbf{m} = \begin{pmatrix} \mathbf{m}_x \\ \mathbf{m}_y \end{pmatrix} \quad \text{and} \quad \mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{xx} & \mathbf{\Sigma}_{xy} \\ \mathbf{\Sigma}_{yx} & \mathbf{\Sigma}_{yy} \end{pmatrix},$$

where $\mathbf{x} \sim N_n(\mathbf{m}_x, \mathbf{\Sigma}_{xx})$ *and* $\mathbf{y} \sim N_p(\mathbf{m}_y, \mathbf{\Sigma}_{yy})$. *Then, the conditional distribution of* $\mathbf{x}$ *given* $\mathbf{y}$ *is also normally distributed where the mean is*

given by

$$\mathbf{m}_{\mathbf{x}|\mathbf{y}} = \mathbf{m}_x + \mathbf{\Sigma}_{xy}\mathbf{\Sigma}_{yy}^{-1}(\mathbf{y} - \mathbf{m}_y), \tag{7.5}$$

and the covariance matrix is given by

$$\mathbf{P}_{\mathbf{x}|\mathbf{y}} = \mathbf{\Sigma}_{xx} - \mathbf{\Sigma}_{xy}\mathbf{\Sigma}_{yy}^{-1}\mathbf{\Sigma}_{yx}. \tag{7.6}$$

Proof. — The proof is given in Eaton (1983), p.116. ∎

This theorem can be applied to each analysis step of the Kalman filter (Algorithm 4.1) with $\mathbf{z} = (\mathbf{x}_i^f, \mathbf{y}_i)^T$. To this aim, we assume that :

$$\mathbf{x}_i^t \sim N(\mathbf{x}_i^f, \mathbf{P}_i^f), \tag{7.7}$$
$$\boldsymbol{\epsilon}_i^o \sim N(0, \mathbf{R}_i), \tag{7.8}$$
$$E\left[(\mathbf{x}_i^t - \mathbf{x}_i^f)(\boldsymbol{\epsilon}_i^o)^T\right] = 0, \tag{7.9}$$
$$\mathbf{M}_{i+1,i} \text{ and } \mathbf{H}_i \text{ are linear}, \tag{7.10}$$

where the Assumption 7.9 states that the error in the forecast vector and in the observations are uncorrelated. The mean of $\mathbf{z}$ is thus given by $\mathbf{m} = (\mathbf{x}_i^f,\ \mathbf{H}_i\mathbf{x}_i^f)^T$ since $E[\mathbf{x}_i^t] = \mathbf{x}_i^f$ by Assumption 7.7 and since

$$\begin{aligned} E[\mathbf{y}_i] &= E[\mathbf{H}_i\mathbf{x}_i^t + \boldsymbol{\epsilon}_i^o] \\ &= \mathbf{H}_i E[\mathbf{x}_i^t] + E[\boldsymbol{\epsilon}_i^o] \\ &= \mathbf{H}_i\mathbf{x}_i^f, \end{aligned}$$

using the definition of the observation $\mathbf{y}_i$ and using the Assumption 7.8. The covariance matrix of $\mathbf{z}$ is given by

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{P}_i^f & \mathbf{P}_i^f\mathbf{H}_i^T \\ \mathbf{H}_i\mathbf{P}_i^f & \mathbf{H}_i^T\mathbf{P}_i^f\mathbf{H}_i + \mathbf{R}_i \end{pmatrix}$$

since

$$\begin{aligned}
\mathrm{Cov}(\mathbf{x}_i^t, \mathbf{y}_i) &= E\left[\left(\mathbf{x}_i^t - E\left[\mathbf{x}_i^t\right]\right)\left(\mathbf{y}_i - E\left[\mathbf{y}_i\right]\right)^T\right] \\
&= E\left[\left(\mathbf{x}_i^t - \mathbf{x}_i^f\right)\left(\mathbf{H}_i\mathbf{x}_i^t + \boldsymbol{\epsilon}_i^o - \mathbf{H}_i\mathbf{x}_i^f\right)^T\right] \\
&= E\left[\left(\mathbf{x}_i^t - \mathbf{x}_i^f\right)\left(\mathbf{H}_i(\mathbf{x}_i^t - \mathbf{x}_i^f) + \boldsymbol{\epsilon}_i^o\right)^T\right] \\
&= E\left[\left(\mathbf{x}_i^t - \mathbf{x}_i^f\right)\left(\mathbf{x}_i^t - \mathbf{x}_i^f\right)^T\mathbf{H}_i^T\right] + E\left[\left(\mathbf{x}_i^t - \mathbf{x}_i^f\right)\left(\boldsymbol{\epsilon}_i^o\right)^T\right] \\
&= E\left[\left(\mathbf{x}_i^t - \mathbf{x}_i^f\right)\left(\mathbf{x}_i^t - \mathbf{x}_i^f\right)^T\right]\mathbf{H}_i^T \\
&= \mathbf{P}_i^f\mathbf{H}_i^T,
\end{aligned}$$

by Assumptions 7.9 and 7.10, and since

$$\begin{aligned}
\mathrm{Cov}(\mathbf{y}_i, \mathbf{y}_i) &= E\left[\left(\mathbf{y}_i - E\left[\mathbf{y}_i\right]\right)\left(\mathbf{y}_i - E\left[\mathbf{y}_i\right]\right)^T\right] \\
&= E\left[\left(\mathbf{H}_i\mathbf{x}_i^t + \boldsymbol{\epsilon}_i^o - \mathbf{H}_i\mathbf{x}_i^f\right)\left(\mathbf{H}_i\mathbf{x}_i^t + \boldsymbol{\epsilon}_i^o - \mathbf{H}_i\mathbf{x}_i^f\right)^T\right] \\
&= E\left[\left(\mathbf{H}_i(\mathbf{x}_i^t - \mathbf{x}_i^f) + \boldsymbol{\epsilon}_i^o\right)\left(\mathbf{H}_i(\mathbf{x}_i^t - \mathbf{x}_i^f) + \boldsymbol{\epsilon}_i^o\right)^T\right] \\
&= E\left[\mathbf{H}_i\left(\mathbf{x}_i^t - \mathbf{x}_i^f\right)\left(\mathbf{x}_i^t - \mathbf{x}_i^f\right)^T\mathbf{H}_i^T\right] + E\left[\boldsymbol{\epsilon}_i^o\left(\boldsymbol{\epsilon}_i^o\right)^T\right] \\
&\quad + 2\mathbf{H}_i E\left[\left(\mathbf{x}_i^t - \mathbf{x}_i^f\right)\left(\boldsymbol{\epsilon}_i^o\right)^T\right] \\
&= \mathbf{H}_i\mathbf{P}_i^f\mathbf{H}_i^T + \mathbf{R}_i.
\end{aligned}$$

by Assumptions 7.8, 7.9 and 7.10. Thus, the conditional distribution of $\mathbf{x}_i^t$ given $\mathbf{y}_i$ follows a multivariate normal distribution of mean

$$\mathbf{m}_{\mathbf{x}|\mathbf{y}} = \mathbf{x}_i^f + \mathbf{P}_i^f\mathbf{H}_i^T(\mathbf{H}_i\mathbf{P}_i^f\mathbf{H}_i^T + \mathbf{R}_i)^{-1}(\mathbf{y}_i - \mathbf{H}_i\mathbf{x}_i^f)$$

and of covariance matrix

$$\mathbf{P}_{\mathbf{x}|\mathbf{y}} = \mathbf{P}_i - \mathbf{P}_i\mathbf{H}_i^T(\mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^T + \mathbf{R}_i)^{-1}\mathbf{H}_i\mathbf{P}_i.$$

This result is equivalent to the Kalman filter correction step presented in Algorithm 4.1 as $\mathbf{m}_{\mathbf{x}|\mathbf{y}} = \mathbf{x}_i^a$ and $\mathbf{P}_{\mathbf{x}|\mathbf{y}} = \mathbf{P}_i^a$. To complete the equivalence with the Kalman filter, one can observe that if an analysis state vector $\mathbf{x}_i^t \sim N_n(\mathbf{x}_i^a, \mathbf{P}_i^a)$, then we have that $\mathbf{M}_{i+1,i}\mathbf{x}_i^t \sim N_n(\mathbf{M}_{i+1,i}\mathbf{x}_i^a, \mathbf{M}_{i+1,i}\mathbf{P}_i^a\mathbf{M}_{i+1,i}^T)$ using the property of multivariate normal distribution (see Montgomery and Runger, 2007, for example). It shows that the propagation of a multivariate normal distribution is equivalent to the forecast step of Algorithm 4.1. Besides this equivalence between the Bayesian approach and the Kalman filter, it is possible to prove that

the mean $\mathbf{m}_{\mathbf{x}|\mathbf{y}}$ defined in (7.5) is the minimal variance estimator among the set of all estimators (Anderson and Moore, 1979, p.26). This result is stronger than the one presented in Subsection 4.1.2 which states that the estimator produced by the Kalman filter is only the BLUE estimator, i.e., the minimal variance estimator among the set of all *linear* estimators.

We now present the two ensemble-based Kalman filters which deal with nonlinear operators using the Bayesian approach and the assumption of multivariate normal distributions.

7.3.1 The unscented Kalman filter

The unscented Kalman filter (UKF) is an ensemble-based Kalman filter introduced by Julier and Uhlmann (1997) and reviewed in Julier and Uhlmann (2004). It uses a deterministic sampling technique known as the unscented transform to generate an ensemble of state vectors. At each iteration, a minimal set of $2n$ sample points (called sigma points) are deterministically picked up around the mean $\mathbf{x}_{i-1}^a$ according to the covariance matrix $\mathbf{P}_{i-1}^a$. These sigma points are then propagated through the nonlinear model operator, from which the mean $\mathbf{x}_i^f$ and covariance $\mathbf{P}_i^f$ are then recovered. For the analysis step a new set of $2n$ sigma points are deterministically picked up around the mean $\mathbf{x}_i^f$ according to the covariance matrix $\mathbf{P}_i^f$. This set is used to determine the covariance matrices involved in (7.5) and (7.6) and allows to compute $\mathbf{x}_i^a$ and $\mathbf{P}_i^a$. The unscented Kalman filter captures the true estimate and covariance matrix accurately to the third order for any nonlinear system (Simon, 2006). Unfortunately, it is too costly for large problems since it implies $2n$ model integrations at each iteration for the propagation of the $2n$ sigma points.

7.3.2 The ensemble Kalman filter

The Ensemble Kalman filter (EnKF) randomly generates an ensemble of state vectors using a Monte Carlo approach (Evensen, 2007). Its advantage is that it requires an ensemble size much lower than the size of the state vector. Indeed, it has been shown that an ensemble size of 50 to 100 is often adequate for systems with thousands of variables (Snyder et al., 2008). For this reason, the EnKF is particularly well suited for large nonlinear data assimilation problems. The original EnKF proposed by Evensen (1994) is based on perturbations of the observations. We briefly explain how it works below (see also Algorithm 7.3). At the initialization phase (line 1), an ensemble of q state vectors is randomly generated, following a Gaussian distribution of mean $\mathbf{x}_b$ and of covariance $\mathbf{B}$. The EnKF filter then assimilates the N observations sequentially. At each iteration i of the loop, a forecast step is first performed, in which each member of the ensemble is integrated up to the next observation time, yielding a forecast ensemble (line 4). The sample mean (line 5) and the sample covariance matrix (line 7) of this ensemble are computed to obtain an estimate of the true

Algorithm 7.3 Ensemble Kalman filter

1: $\left\{\mathbf{x}^a(j) = \mathbf{x}^b + \boldsymbol{\epsilon} \;\mid\; \boldsymbol{\epsilon} \sim N(0, \mathbf{B}), \quad j = 1, \ldots, q\right\}$
2: **for** $i = 1$ to N **do**
3: (* **Forecast step** *)
4: $\left\{\mathbf{x}^f(j) = \mathcal{M}_{i,i-1}\mathbf{x}^a(j) \;\mid\; j = 1, \ldots, q\right\}$
5: $\bar{\mathbf{x}}^f = \dfrac{1}{q}\displaystyle\sum_{j=1}^{q} \mathbf{x}^f(j)$
6: $\mathbf{X}^f = \left[\mathbf{x}^f(1) - \bar{\mathbf{x}}^f, \ldots, \mathbf{x}^f(q) - \bar{\mathbf{x}}^f\right]$
7: $\bar{\mathbf{P}}^f = \dfrac{1}{q-1}\mathbf{X}^f(\mathbf{X}^f)^T$
8: (* **Correction step** *)
9: $\left\{\mathbf{y}^f(j) = \mathcal{H}_i\mathbf{x}^f(j) \;\mid\; j = 1, \ldots, q\right\}$
10: $\bar{\mathbf{y}}^f = \dfrac{1}{q}\displaystyle\sum_{j=1}^{q} \mathbf{y}^f(j)$
11: $\mathbf{Y}^f = \left[\mathbf{y}^f(1) - \bar{\mathbf{y}}^f, \ldots, \mathbf{y}^f(q) - \bar{\mathbf{y}}^f\right]$
12: $\mathbf{K} = \mathbf{X}^f(\mathbf{Y}^f)^T(\mathbf{Y}^f(\mathbf{Y}^f)^T + \mathbf{R}_i)^{-1}$
13: $\left\{\mathbf{y}_i(j) = \mathbf{y}_i + \boldsymbol{\epsilon}_i | \boldsymbol{\epsilon}_i \sim N(0, \mathbf{R}_i), j = 1, \ldots, q\right\}$
14: $\left\{\mathbf{x}^a(j) = \mathbf{x}^f(j) + \mathbf{K}(\mathbf{y}_i(j) - \mathbf{y}^f(j)) \;\mid\; j = 1, \ldots, q\right\}$
15: $\bar{\mathbf{x}}^a = \dfrac{1}{q}\displaystyle\sum_{j=1}^{q} \mathbf{x}^a(j)$
16: $\mathbf{X}^a = \left[\mathbf{x}^a(1) - \bar{\mathbf{x}}^a, \ldots, \mathbf{x}^a(q) - \bar{\mathbf{x}}^a\right]$
17: $\bar{\mathbf{P}}^a = \dfrac{1}{q-1}\mathbf{X}^a(\mathbf{X}^a)^T$
18: **end for**

state and of its error covariance matrix, respectively, before correction by the observations. Since the size of the ensemble is lower than the size of the state vector, the error covariance matrix is of low rank. At the correction step, the counterpart of the forecast ensemble generated on line 4 is mapped into the observation space (line 9) and a Kalman gain matrix $\mathbf{K}$ is derived (line 12). An ensemble of q perturbed observations is then randomly generated, using a Gaussian distribution of mean $\mathbf{y}_i$ and covariance $\mathbf{R}_i$ (line 13). Each member of the forecast ensemble is finally corrected by the Kalman gain using a perturbed observation (line 14), hence producing the correction ensemble. The sample mean (line 15) and the sample covariance matrix (line 17) of this correction ensemble is then computed to get an estimate of the true state and

of its error covariance matrix after correction by the observations. Note that the sample covariance matrices (lines 7 and 17) are computed for diagnostic reasons. In an operational framework, the computation of the Kalman gain can be made less expensive (see Evensen, 2003).

Besides the fact that this filter does not require Jacobians computation and is suited for large systems, the EnKF has a simple formulation, is relatively easy to implement and naturally parallelizable. For these reasons, it has gained popularity. Moreover, the estimate of the error covariance matrix evolves dynamically during the assimilation process. Other advantages are developed in Kalnay et al. (2007) and Lorenc (2003), where a comparison with the 4D-Var is performed. The EnKF community is very active, proposing a lot of improvements and variants of the initial algorithm. For examples, Evensen (2004) examines how different sampling strategies and implementations of the correction scheme influence the performance of the EnKF, while in Ott et al. (2004) and Hunt et al. (2007), a local EnKF has been developed where the correction is performed grid point by grid point using observations on a local region. An approach also exists which determines a square root Kalman gain without perturbing the observations (see Whitaker and Hamill (2002)).

To our knowledge, there are no method for the 4D-Var formulation which do not rely on the Jacobian information. Before trying to fill this gap, we shortly review the current state of the art in the derivative-free optimization community.

7.4 Derivative-free optimization

Nonlinear optimization methods generate a sequence of iterates designed to asymptotically converge to a solution. The best way to build this sequence is to rely on the information given by the (usually first and second-order) derivatives computed at the current iterate, to cleverly produce a step to a good, next iterate. This is the basis of Newton or quasi-Newton methods which have been presented in Section 6.1. When derivatives are difficult or costly to compute, unreliable or even unavailable, approximating them by finite differences is worth being used in some applications but cannot be regarded as a general technique, because of the excessive number of function evaluations it may need and the possible inaccuracy it may induce. Under such circumstances, one can use Derivative-Free Optimization (DFO) methods.

There are three main classes of DFO algorithms. Two of them are deterministic (Conn et al., 2009) and one is stochastic. The first deterministic DFO class of methods maintains linear or quadratic approximations[1] of the functions involved in the problem by constructing them using only function evaluations

[1] To avoid confusion with *oceanic or athmospheric models*, we use the word *approximation* to denote a surrogate of a given function, instead of the word *model* commonly used in the optimization community.

computed at appropriate sample points. These approximations are built using polynomial interpolations or regressions, or by any other approximation technique. The power of this class of DFO algorithms comes from the trust-region framework (Section 6.1) which is well-suited for nonlinear optimization problems. The second class of deterministic DFO algorithms is named *direct-search methods*. It directly exploits a sample set of function values without building an explicit approximation. One of the simplest direct-search method is called *coordinate search*. In an unconstrained framework, it evaluates, from a current iterate $\mathbf{x}_j$, with a current step size parameter α_j, and following a specific order, the function to minimize, f say, in a *poll set* P_j defined by

$$P_j = \{\mathbf{x}_j + \alpha_j \mathbf{d} : \mathbf{d} \in \mathbf{D}\},$$

where $\mathbf{D} = [\mathbf{e}_1, \ldots, \mathbf{e}_n, -\mathbf{e}_1, \ldots, -\mathbf{e}_n]$ and $\mathbf{e}_i$, $i = 1, \ldots, n$, are the vectors of the canonical basis. When a point $\mathbf{x}_j + \alpha_j \mathbf{d}$ decreases the function value, it is selected as the next iterate, setting $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{d}$, and the step size is increased. If none of the points in P_j leads to a decrease in the function value, the step size parameter is shrunk and $\mathbf{x}_{j+1} = \mathbf{x}_j$. The basic steps of the coordinate-search method are summarized in Algorithm 7.4. Figure 7.1

Algorithm 7.4 Coordinate-search method

Choose $\mathbf{x}_0$ and α_0
Set $0 < \gamma_1 < 1$ and $\gamma_2 \geqslant 1$
for $k = 0, 1, 2, \ldots$ **do**
  $P_j = \{\mathbf{x}_j + \alpha_j \mathbf{d} : \mathbf{d} \in \mathbf{D}\}$ (poll set)
  Evaluate f at the poll points using a given order
  if for some poll point $f(\mathbf{x}_j + \alpha_j \mathbf{d}) < f(\mathbf{x}_j)$ **then**
    Set $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{d}$ (successful iteration)
    Set $\alpha_{j+1} = \gamma_2 \alpha_j$
    Stop evaluating
  else
    Set $\mathbf{x}_{j+1} = \mathbf{x}_j$ (unsuccessful iteration)
    Set $\alpha_{j+1} = \gamma_1 \alpha_j$
  end if
end for

illustrates the coordinate-search method on a simple two variables function for the parameter values $\gamma_1 = 0.5$ and $\gamma_2 = 1$, using the order "left, up, right, down" to evaluate the poll points. The circles represent poll points at which the objective function has been evaluated while the squares represent the iterates. Coordinate-search methods belong to a class of so-called *pattern-search methods*[2] whose convergence, for the interested reader, is analyzed and

[2] Pattern-search methods form a subclass of the direct-search methods, see Lewis et al. (2000) for a classification tree of direct-search methods.

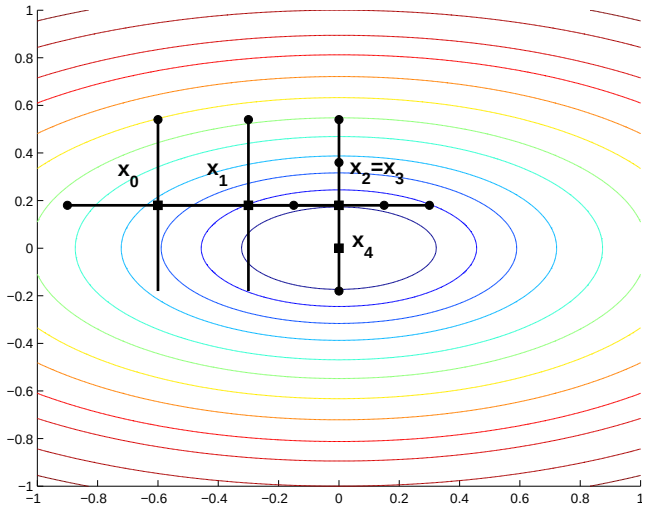

Figure 7.1 — *The coordinate-search iterates for* $f(x, y) = x^2 + 5y^2$.

guaranteed (without prematurely termination due to inadequate step length control mechanisms) in (Torczon, 1993). The third class of DFO methods is based on random choices and includes the evolutionary family, among which the well-known genetic algorithms (Ashlock, 2005).

To our purpose, we have chosen a deterministic method of direct-search type to illustrate the numerical performance of our DFO approach (see Sections 7.7 and 7.10). The direct-search methods have some advantages compared to DFO methods based on approximations. Indeed, evaluating the 4D-Var function in (3.6) requires a PDE integration to evaluate $\mathcal{G}(\mathbf{x})$, which may fail to work. Direct-search methods are best suited to easily face this problem by simply ignoring the problematic point and skipping it (Booker et al., 1998). Moreover, direct-search methods are more adapted to be implemented on parallel machines, taking advantage of any number of processors (Dennis and Torczon, 1991). The main idea is to add more directions in the poll set P_j and to perform multiple searches along these directions simultaneously. On the contrary, when using approximation-based DFO techniques, parallelization of the linear algebra steps in Newton or quasi-Newton like methods is limited to only few processors (Byrd et al., 1988, Nash and Sofer, 1989).

We are now ready to present the different components of our DFO approach to solve the 4D-Var problem (3.6). Before that, it is important to notice that, despite the huge efforts in the development of DFO methods by the optimization community, it is clear that there are still considerable disadvantages not having or using the derivative information. One cannot expect the performance of DFO methods to be comparable to that of Newton or quasi-Newton like methods based on derivatives. It is also more challenging to decide when

convergence has occurred and to design stopping criteria when no derivatives are available. Another major drawback of DFO methods is related to the size of the problem to solve. It is usually not reasonable to use them to solve medium or large problems, i.e., problems with more than a few tens of variables, because of their need of function evaluations that becomes quickly not affordable. To overcome this drawback, the usual remedy is to attempt to determine the most critical variables involved in the problem and to optimize on these only, hence drastically reducing the dimension of the search space.

7.5 Reducing the dimension of the problem

The computational cost of solving the 4D-Var problem (3.6) can be reduced in several ways: through the model integration, the reduction of the number of function evaluations or the reduction of the state space dimension (Biegler et al., 2011). From a DFO point of view, the crucial issue is the reduction of the state space dimension.

Clearly, in three dimensions, discretizing PDEs can rapidly lead to millions of variables. However, quite often, the way the state vector is organized (in smoothness and/or in structure) allows to identify subspaces containing the essential of the variability of the problem. A first possibility to reduce the dimension of the state space is based on the Empirical Orthogonal Functions (EOFs) developed for the SEEK filter presented in Subsection 4.3.3. Another possibility to reduce the dimension of the state space is to use balanced truncation (Antoulas (2005), see also Moore (1981) and Hahn and Edgar (2002) for the linear and the nonlinear case, respectively). Both methods compute eigenvectors and rely on modal truncation. It means that the decrease of the eigenvalues is used to perform the truncation since the essential of the variability of the problem is contained in the subspace spanned by the eigenvectors corresponding to the largest eigenvalues. No universal rule exists but they all amount to fix a threshold r on the number of eigenvectors used to build the reduced space (see, criteria 4.31, for example). These eigenvectors are stored in a matrix $\mathbf{L} \in \mathbb{R}^{n \times r}$ with the property that $\mathbf{L}^T\mathbf{L} = I_r$. Once the matrix $\mathbf{L}$ has been computed, one can focus on the search of a solution of the type $\mathbf{x} = \mathbf{x}^b + \mathbf{L}\delta\underline{\mathbf{x}}$, where $\delta\underline{\mathbf{x}} \in \mathbb{R}^r$ is an approximate solution of

$$\min_{\delta\underline{\mathbf{x}} \in \mathbb{R}^r} \frac{1}{2}||\mathbf{L}\delta\underline{\mathbf{x}}||^2_{\mathbf{B}^{-1}} + \frac{1}{2}||\mathcal{G}(\mathbf{x}^b + \mathbf{L}\delta\underline{\mathbf{x}}) - \mathbf{y}||^2_{\mathbf{R}^{-1}}, \tag{7.11}$$

which is a reduced form of the 4D-Var problem (3.6) in the subspace generated by $\mathbf{L}$. In the following, the reduction technique based on the EOFs is adopted since we have had a positive experience using them in the previous chapter.

7.6 Iterative subspace minimization

In operational ocean models such as OPA, a typical size for the set of state vectors used in the EOFs analysis is equal to $m = 500$ and the number of computed EOFs is equal to $r = 50$ (Hoteit and Pham, 2003). In a DFO context however, a crucial issue is the number of variables, hence the choice of the subspace dimension for our concern. Indeed, if this subspace is too large, solving the problem could be far too expensive, while if this subspace is too small, part of the important variability could be missed. For these reasons, we propose an approach which allows as much flexibility as possible. This approach builds a sequence of subspaces with appropriate size with respect to the use of a DFO method, while exploiting as much as possible the information contained in the computed EOFs.

Let us thus consider s subspaces generated by the matrices $\mathbf{L_k} \in \mathbb{R}^{n \times r_k}$, $k = 1, \ldots, s$ (we propose below two strategies to choose these subspaces, based on different hierarchies of the EOFs). Our method basically computes a sequence of approximations of the solution of the 4D-Var (3.6) using the update formula $\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{L}_k \delta \underline{\mathbf{w}}_k$, where $\delta \underline{\mathbf{w}}_k$ minimizes

$$\min_{\delta \underline{\mathbf{w}} \in \mathbb{R}^{r_k}} \underline{J}_k(\delta \underline{\mathbf{w}}) = \frac{1}{2} ||\mathbf{L_k} \delta \underline{\mathbf{w}} - (\mathbf{x}^b - \mathbf{x}_k)||^2_{\mathbf{B}^{-1}} + \frac{1}{2} ||\mathcal{G}(\mathbf{x}_k + \mathbf{L}_k \delta \underline{\mathbf{w}}) - \mathbf{y}||^2_{\mathbf{R}^{-1}}. \quad (7.12)$$

This technique, that we call *iterative subspace method (ISM)*, is summarized in Algorithm 7.5. As already mentioned, the matrices $\mathbf{L}_k$ can be generated

Algorithm 7.5 Iterative subspace method (ISM)

1: $\mathbf{x}_0 = \mathbf{x}^b$
2: Compute m EOFs using (4.30) and choose a hierarchy
3: **for** $k = 1, \ldots, s$ **do**
4: Construct $\mathbf{L}_k$ according to the hierarchy
5: Use a coordinate-search algorithm to minimize (7.12)
6: Set $\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{L}_k \delta \underline{\mathbf{w_k}}$
7: **end for**

using other tools than the EOFs and any DFO algorithm could be used to solve (7.12). An important issue remains the choice of the dimensions r_k for the iterated subspaces. To some extend, this choice may be considered as dependent on how computer ressources and DFO algorithms evolve. We are now ready to illustrate our ISM algorithm on a two dimensional shallow water data assimilation problem.

7.7 Results on a shallow water model

Consider the shallow water equations in a two dimension Cartesian coordinate system

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - fv + g\frac{\partial z}{\partial x} &= 0, \\ \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + fu + g\frac{\partial z}{\partial x} &= 0, \\ \frac{\partial z}{\partial t} + \frac{\partial}{\partial x}[u(z-b)] + \frac{\partial}{\partial y}[v(z-b)] &= 0, \end{aligned}$$

where u is the zonal velocity, v is the meridional velocity, z is the height field, b is the height of the ground, g is the gravity and f is the Coriolis parameter induced by the rotation of the earth (Figure 7.2).

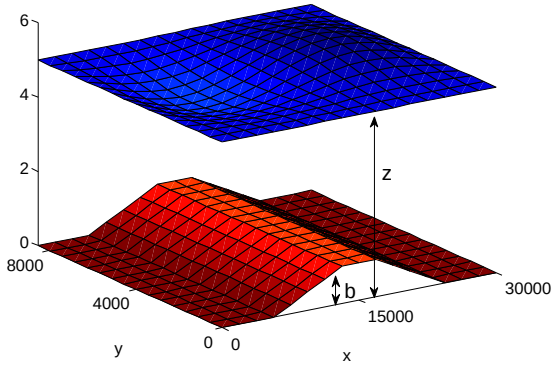

Figure 7.2 — *Shallow water model.*

The results we present in this section have been obtained using a Matlab implementation of these equations developed by S. Gratton, Ph. Toint, J. Tshimanga, S. Gurol and O. Titaud. In their Matlab data assimilation framework, the authors have chosen a discretization that uses a uniform grid and a Leapfrog scheme is applied to integrate the PDEs. Dirichlet boundary conditions are imposed by specifying the solutions on the boundary of the domain along y, while periodic boundary conditions are applied on the boundary of the domain along x. Tangent and adjoint codes are generated by automatic differentiation with a backward mode. Our DFO approach will not use these codes but they will be useful for diagnostic and comparative reasons.

The framework for the assimilation problem is the classical twin experiment. The true trajectory is computed from a true initial condition. The observations, distributed in space and time, are extracted from this trajectory by adding an observation noise. Only the height field z is observed. The background is

equal to the true initial condition where a phase and amplitude error has been added. For our numerical illustration, we consider five different experiments on increasingly finer grids. The size of the state vector varies from 816 to 197376, corresponding to grids of sizes 16×17, 32×33, 64×65, 128×129 and 256×257, respectively. The values for the most important parameters (the size of the state vector n and the total number of observations p) for each experiment are summarized in Table 7.1.

#	n	p	Assimilation window length
1	816	48	24 hours
2	3168	896	24 hours
3	12480	3584	24 hours
4	49536	504	12 hours
5	197376	1008	12 hours

Table 7.1 — *Parameters values for the 5 different experiments.*

Referring to Algorithm 7.5, we first need to compute EOFs. These are computed from a set of $m = 100$ state vectors collected during 20 days. The singular values corresponding to these 100 EOFs are plotted in Figure 7.3 for the fourth experiment. The first (obvious) hierarchy we consider consists in sorting the EOFs following a decreasing order of their corresponding singular values. Each matrix $\mathbf{L}_k$ is then constructed using a batch of $r_k = 10$ successive EOFs, starting from the batch corresponding to the 10 largest singular values. For each subspace minimization, we use a Matlab implementation of Algorithm 7.4 above (with some improvements) called SID-PSM (Custodio et al., 2010, Custodio and Vicente, 2007) to approximately solve (7.12). The stopping criterion for the SID-PSM code is based on a minimal value for the step size parameter. The algorithm stops if this step size parameter becomes smaller than a threshold value, fixed to 10^{-3} in our experiments. Finally, to mimic the solution of real-life data assimilation problems which are real-time optimization problems, we put a limit on the computational time budget. In our framework, we have fixed a limit of 1000 evaluations of the objective function $\underline{J}_k$ in (7.12). This limit, together with the step size threshold used in SID-PSM, are the unique stopping criteria, making possible that only a subset of the $s = 10$ subspaces generated by the matrices $\mathbf{L}_k$ are explored.

The results of the ISM Algorithm 7.5 on the shallow water problem are presented in Figure 7.4 and illustrate the behavior of the method on our set of five experiments. The plots represent the 4D-Var function value versus the number of function evaluations. Each dot on the graphs means that the step size threshold has been reached in SID-PSM and a new subspace is explored. For the four first experiments, five subspaces are investigated, while only four are for the fifth experiment. A way to improve the quality of the solution obtained within the allowed budget is clearly to improve the quality of the

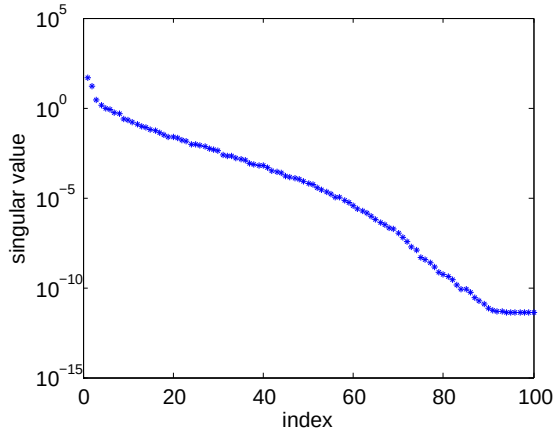

Figure 7.3 — *Singular values (n=49536).*

successive subspaces used. We devote the next sections to this goal, along two axes. The first improves the quality of the EOFs and the second proposes another EOFs hierarchy for the construction of the iterated subspaces.

7.8 Improving the EOFs

As mentioned in Section 4.3.3, the EOFs are computed from a set of state vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{R}^n$ supposedly representative of the variability of the system and computed on a time interval before the assimilation window. In practice, the model is integrated over a long period from a previous state estimate. Since successive states are quite similar, only m of them are selected at regular frequencies. This procedure does not need to be restarted for each new data assimilation window as the information can be used during a certain time.

The closer the state vectors selected for the EOFs computation are to the true trajectory, the better is the quality of the subspace generated by the computed EOFs. To illustrate this relation, we consider a time interval of 20 days much prior to the assimilation window and denote by t^* the initial time of this time interval. We generate a set of 100 equally distributed state estimates x^i using the formula

$$\mathbf{x}^i = \mathbf{x}^t(t^*) + \rho_i(\mathbf{x}^b(t^*) - \mathbf{x}^t(t^*)),$$

where $\mathbf{x}^t(t^*)$ and $\mathbf{x}^b(t^*)$ are the true state and the background, respectively, at time t^*, and $\rho_i \in [0, 1]$. Using each state estimate $\mathbf{x}^i$ as initial condition, a model integration is performed for 20 days, storing $m = 100$ state vectors, from which a set of 100 EOFs are computed to form the basis of a subspace $\mathbf{L}$. A subspace will be considered as of high quality if the minimization of

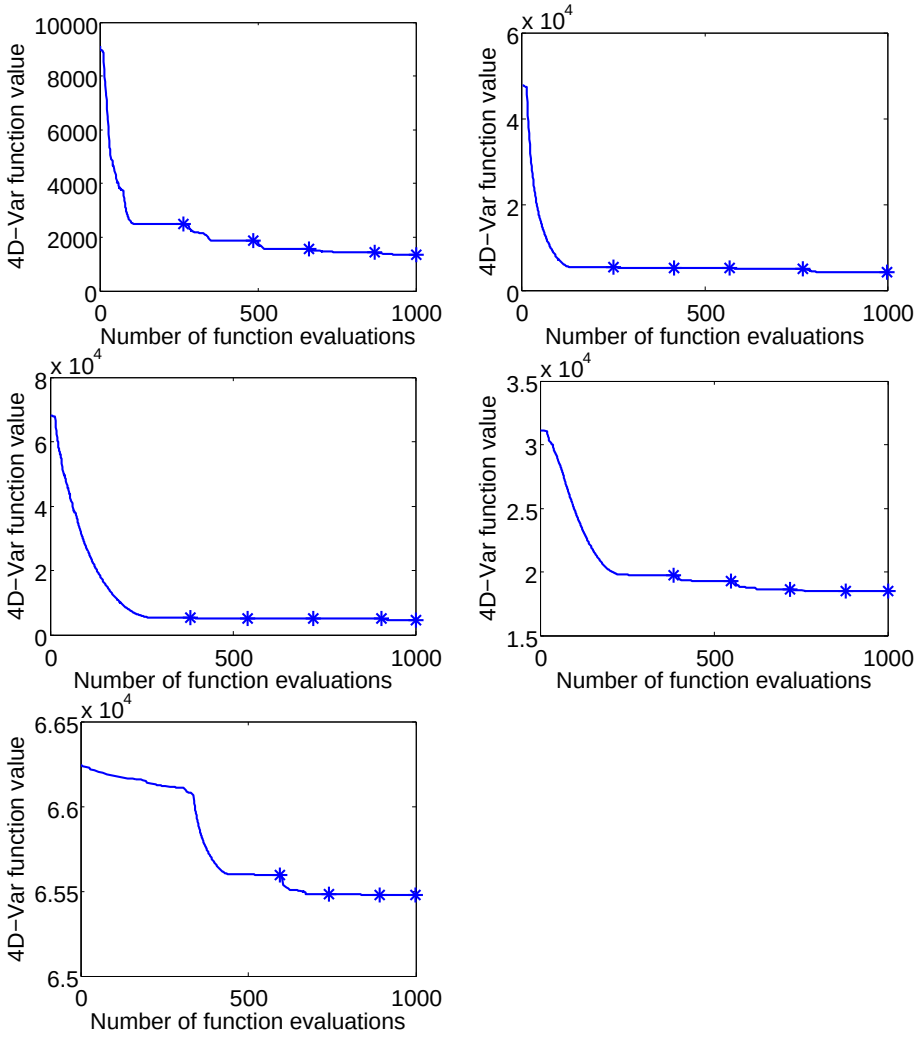

Figure 7.4 — *Results of the iterative subspace method on the shallow water model. Experiment 1 (top left), experiment 2 (top right), experiment 3 (center left), experiment 4 (center right) and experiment 5 (bottom left).*

the reduced 4D-Var (7.11) gives a low function value and of low quality if this minimization gives a high function value. We have used a Gauss-Newton method until convergence to solve (7.11) for each of the 100 subspaces, in order to reach the best possible decrease in the function.

Figure 7.5 shows the minimum function values versus the parameter ρ for

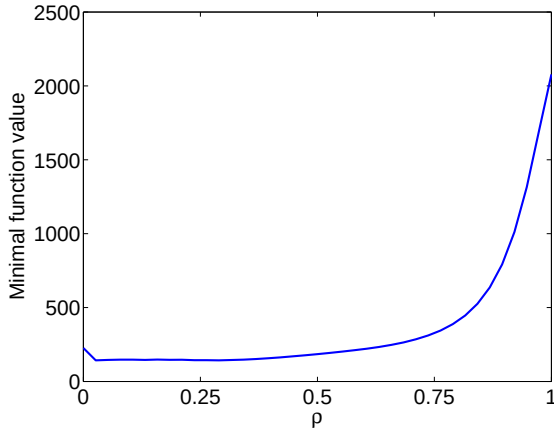

Figure 7.5 — *Minimum function value versus distance to the truth (for experiment 3).*

the third experiment (similar behaviors are observed for the other experiments). When $\rho_i = 0$, the starting point for the EOFs computation is the true state $\mathbf{x}^t(t^*)$, while for $\rho_i = 1$, it is the background $\mathbf{x}^b(t^*)$. We see the importance of selecting state vectors which are close to the true trajectory to obtain subspaces of good quality. However, for operational data assimilation problems without twin experiments, such vectors are hard to select.

Another strategy to increase the quality of the subspace $\mathbf{L}$ is based on smoothing. While the first EOFs corresponding to the largest singular values are smooth, the last ones corresponding to the smallest singular values are generally oscillating. Figure 7.6 shows the height field components (water level) of the first, smooth, EOF and of the 20th, more oscillating, one. The increment

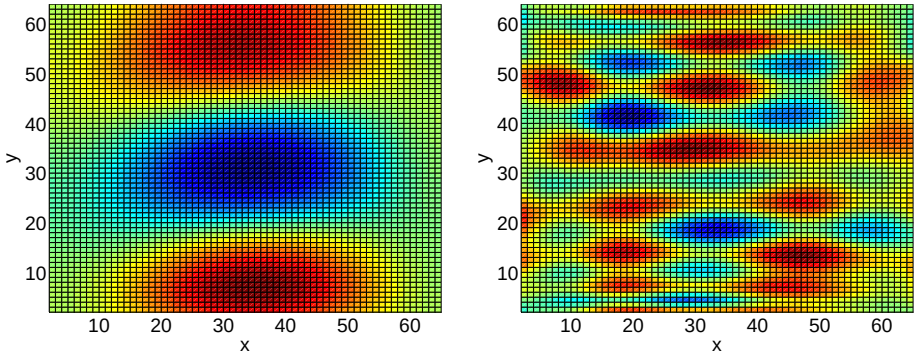

Figure 7.6 — *EOF number 1 (left) and EOF number 20 (right) .*

$\mathbf{L}\delta\underline{\mathbf{x}}$ computed in (7.11) being a linear combination of the m EOFs, it may suffer from the oscillating behavior of the last ones. Frequency filtering could be used to smooth the EOFs and is the basic idea that we develop next.

We first consider a smoothing based on Fourier transformations. To explain this smoothing, consider the three fields of each EOF (u, v and z) represented as images of pixels (see for instance Figure 7.6 for the height field z). These representations define three matrices of size $nx \times ny$, one for each field, where $nx \times ny$ is the number of grid points. Let $\mathbf{I}$ be one of these matrices, if one applies a two-dimensional Fast Fourier Transform (FFT) to this matrix, a new matrix $\mathbf{F} \in \mathbb{R}^{nx \times ny}$ is obtained which contains the frequencies of the corresponding image (see for example the frequency domain drawn for the height field of the 20th EOF in Figure 7.7, where the lowest frequencies are located in the center of the picture and the colors show their amplitude). The smoothing

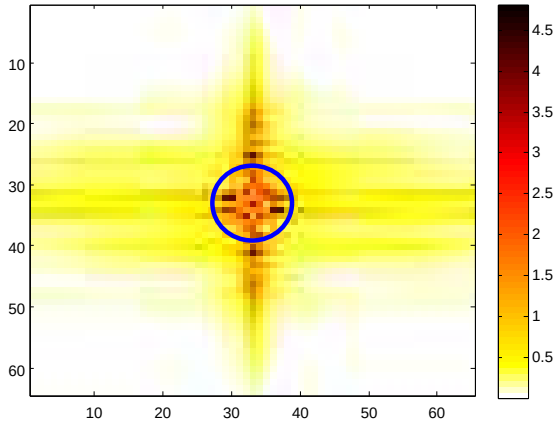

Figure 7.7 — *EOF number 20 in the frequency domain.*

idea consists in defining a *low-pass filter* whose effect suppresses all frequencies higher than a fixed cut-off frequency f_c, leaving all frequences below f_c unchanged. Such a low-pass filter can be defined by a matrix $\mathbf{H} \in \mathbb{R}^{nx \times ny}$ with entries

$$\mathbf{H}_{ij} = \begin{cases} 1, & \text{if } (i - \frac{nx}{2})^2 + (j - \frac{ny}{2})^2 \leqslant f_c \frac{nx^2+ny^2}{4}, \\ 0, & \text{otherwise.} \end{cases}$$

The resulting filtered domain is given by the matrix $\mathbf{G} \in \mathbb{R}^{nx \times ny}$ whose entries are

$$\mathbf{G}_{ij} = \mathbf{F}_{ij} * \mathbf{H}_{ij},$$

where $*$ denotes the multiplication element by element, and corresponds to Figure 7.7 where only the frequencies within a disk centered at $(\frac{nx}{2}, \frac{ny}{2})$ and of radius $\sqrt{f_c \frac{nx^2+ny^2}{4}}$ are retained (see the blue circle, corresponding to a cut-off frequency $f_c = 0.1$). Note that if $f_c = 0$, no frequency is retained, while if

$f_c = 1$, all are kept. In order to go back to the spatial domain, an Inverse Fast Fourier Transform (IFFT) is applied to the matrix $\mathbf{G}$, yielding the EOF's corresponding smoothed field (see Figure 7.8 where the smoothed height field of the 20th EOF is represented for a cut-off frequency value $f_c = 0.1$). If we

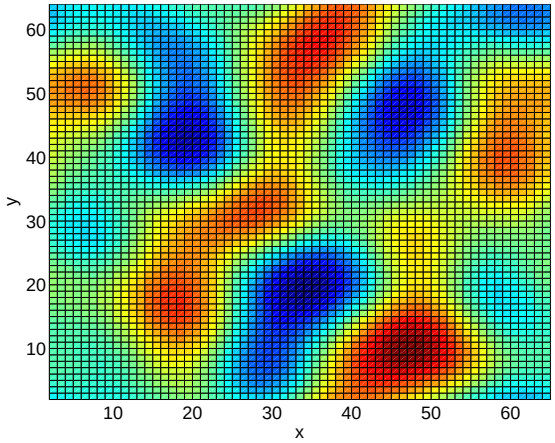

Figure 7.8 — *EOF number 20 smoothed by FFT.*

compare this EOF before and after smoothing, we see that the high frequencies have disappeared while roughly keeping the general shape of the EOF. The main issue is clearly the choice of the parameter f_c. If f_c is too small, the general shape of the EOF will be lost while if f_c is too large, the oscillatory effects will stay.

Figure 7.9 illustrates the quality of the subspace with respect to the value of the parameter $f_c \in [0, 1]$. More precisely, it shows the minimum function values obtained in the smoothed subspace versus the parameter f_c for the second and fourth experiment, where again we have used a Gauss-Newton method until convergence to solve (7.11). One can see that applying a smoothing to the EOFs with a well-chosen f_c increases the quality of the subspace (see the intervals of successful values for f_c), but a bad value for f_c can also damage the quality of the subspace. The different behavior for the two selected experiments also show that it is difficult to choose an optimal or even appropriate value for f_c without sampling part of or the whole set of possible values.

We next consider a smoothing based on the background error covariance matrix $\mathbf{B}$. Usually, this covariance matrix is designed using correlation models (see Weaver and Courtier, 2001), derived from the diffusion equation. That is, the action of this covariance matrix on a vector $\mathbf{x}$ is computed by time-stepping a discretized version of a diffusion equation with initial condition $\mathbf{x}$ over a pseudo-time interval (Weaver and Ricci, 2003, Mirouze and A.Weaver, 2010). This integration can be done explicitly or implicitly (Mirouze and Weaver, 2010). Since integrating a diffusion equation has a smoothing effect on the

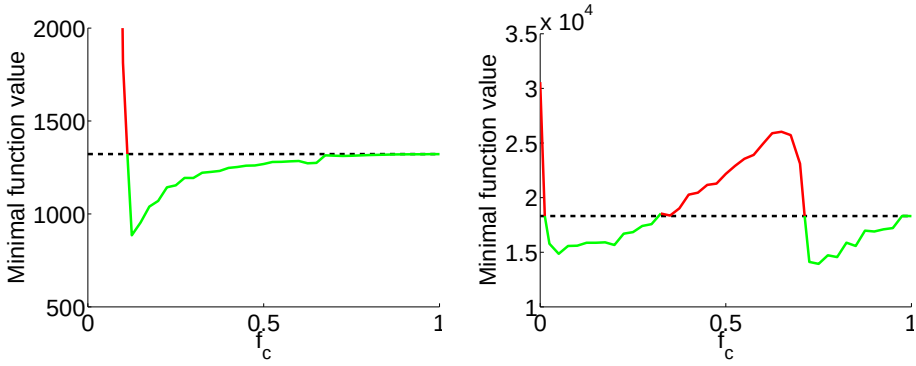

Figure 7.9 — *Minimum function value versus f_c. Experiment 2 (left), experiment 4 (right).*

initial condition, applying a background covariance matrix to a vector can be interpreted as a smoothing (Zhang and Hancock, 2008, Chung, 2006). Note that the background error covariance matrix also depends on a covariance length-scale parameter, but this one is carefully chosen by the modelers to obtain a desirable regularization effect in the 4D-Var in accordance with the application. It thus has an appropriate smoothing strength. The smoothing of the 20th EOFs by $\mathbf{B}$ (which amounts to multiply this EOF by $\mathbf{B}$) is illustrated in Figure 7.10. Here the background error covariance matrix is modeled using a diffusion operator which is integrated using an implicit Euler scheme.

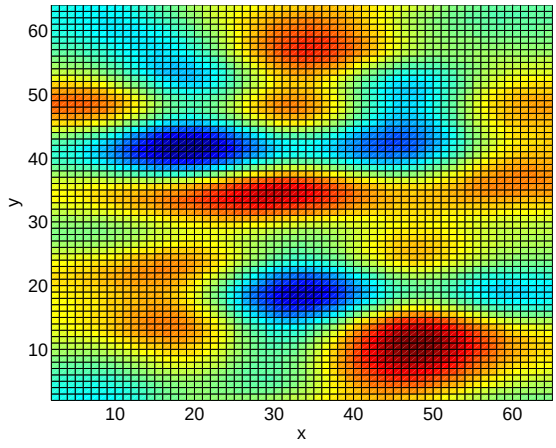

Figure 7.10 — *EOF number 20 smoothed by* $\mathbf{B}$.

Table 7.2 compares the quality of the subspace generated by the EOFs either unsmoothed or smoothed by Fourier transformations (with optimal value for

the parameter f_c) or by the background error covariance matrix. It reports the minimum function values obtained in the (smoothed) subspace when using a Gauss-Newton method until convergence to solve (7.11). Smoothing the EOFs clearly increases, sometimes drastically, the quality of the subspace. The smoothing by $\mathbf{B}$ clearly outperforms the smoothing by Fourier transformations, while being parameter free. We will use this smoothing for the numerical experiments presented in the next sections.

n	minimal function value		
	no smoothing	FFT	**B**
816	142.58	62.22	44.18
3168	1320.03	885.23	756.84
12480	2049.84	1730.87	1470.40
49536	17383.90	13932.23	837.29
197376	66246.88	66245.52	8905.30

Table 7.2 — *Minimal function values for the different smoothing approaches.*

7.9 A new EOFs hierarchy

The EOFs hierarchy we used in our experiments for Section 7.7 were exclusively based on the decreasing order of their corresponding singular values. This criterion may be considered as exploiting the model operator information since the EOFs and the singular values are extracted from a sample covariance matrix built from a set of state vectors.

We now propose a criterion to sort the EOFs based not only on the model operator information but also on the observation operator which arises in (3.7). Indeed, to favour possible improvements when solving the 4D-Var (3.6), it could be interesting to detect the EOFs along which the function $\mathcal{G}$ varies the most. To this goal, let $\mathbf{l}_i$, $i = 1, \ldots, m$, denote the EOFs, with σ_i, $i = 1, \ldots, m$, their corresponding singular values (see (4.22) and (4.30)), we compute the differences

$$\tau_i = ||\mathcal{G}(\mathbf{x}_0) - \mathcal{G}(\mathbf{x}_0 + \mathbf{l}_i)||_{\mathbf{R}^{-1}},$$

for $i = 1, \ldots, m$, which measure the variation of the model $\mathcal{G}$ along each EOF, and rank the EOFs following a decreasing order of the quantities

$$\pi_i = \tau_i + \gamma \, \frac{\max_i \tau_i}{\max_i \sigma_i} \, \sigma_i, \tag{7.13}$$

where $\frac{\max_i \tau_i}{\max_i \sigma_i}$ is a scaling factor and $\gamma > 0$ is a weighting factor. This criterion follows the philosophy developed in Lieberman et al. (2010), with the greedy sampling. Note that since the operator $\mathcal{G}$ is not highly nonlinear, the distances

τ_i do not depend too much on the state vector $\mathbf{x}_0$. It thus makes sense to use the computed hierarchy all along Algorithm 7.5.

Figure 7.11 shows the values of $\pi_i, i = 1, \ldots, 100$, with $\gamma = 5$, for each (**B**-smoothed) EOF $\mathbf{l}_i, i = 1, \ldots, 100$, for the five different experiments. Choosing

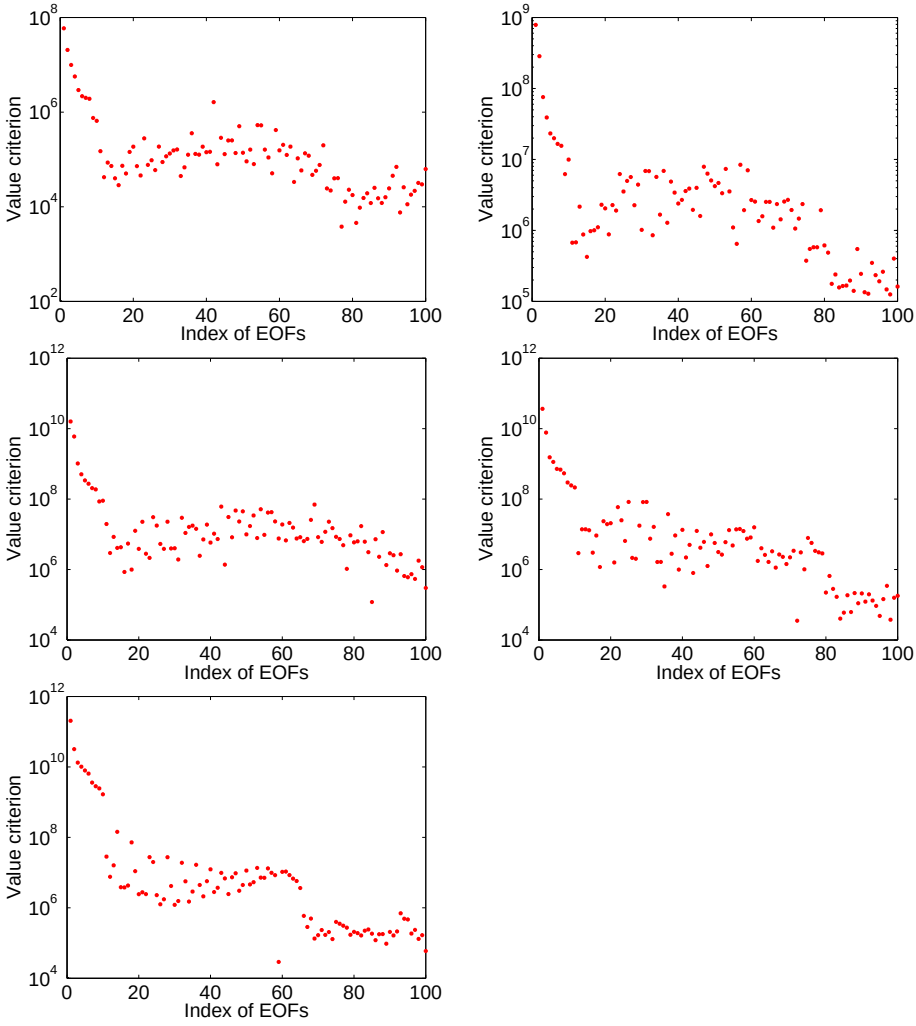

Figure 7.11 — *Criterion values for each EOF. Experiment 1 (top left), experiment 2 (top right), experiment 3 (center left), experiment 4 (center right) and experiment 5 (bottom left).*

a relatively large value for γ emphasizes the importance of the very first EOFs associated to the largest singular values (see the left part of each picture). On one hand, this allows to still favour the EOFs associated to the largest

singular values to build the *first* subspace in Algorithm 7.5. On the other hand, compared with the "natural" hierarchy, the π_i-hierarchy allows to switch faster to subspaces built using EOFs with a larger index (typically between 30 and 70 for the first three experiments), to which correspond larger variations of $\mathcal{G}$.

7.10 Numerical illustration

Coming back to the DFO approach and the iterated subspace method outlined in Section 7.6, we illustrate and compare in this section the behavior of Algorithm 7.5 when considering the two modifications proposed in the previous sections. We first consider the separated effect of adding a smoothing by the bachground matrix $\mathbf{B}$ and of using the EOFs hierarchy based on (7.13). We next illustrate their cumulated effect. We consider exactly the same experimental framework as in Section 7.7, using the same values for the parameters, in particular the same limit of 1000 evaluations of the objective function $\underline{J}_k$ in (7.12). Figure 7.12 shows the behavior of the basic method (solid lines, corresponding to Figure 7.4), the basic method with $\mathbf{B}$-smoothing (dotted lines), the basic method with the π_i-hierarchy (dot-dashed lines) and the basic method with $\mathbf{B}$-smoothing and the π_i-hierarchy (dashed lines) on our set of five experiments. Again, each dot on the graphs means that the step size threshold has been reached in SID-PSM and a new subspace is explored. We first comment on the numerical performance of each version before comparing the cost of each modification. On the first three experiments, one clearly sees that each modification improves the performance individually, while the cumulated effect brings a further improvement. For the two last, larger, experiments, only the smoothing by $\mathbf{B}$ improves the function decrease, while using the EOFs hierarchy based on (7.13) does not improve the subspaces compared with the natural hierarchy. This explains why the cumulated effect of the two modifications is (quasi) similar to that of using only the smoothing by $\mathbf{B}$.

Considering now the cost of implementing each modification, the application of $\mathbf{B}$ to an EOF is cheap compared to a function evaluation, so that smoothing by $\mathbf{B}$ can be considered as a negligible operation. The cost of using the EOFs hierarchy based on (7.13), however, is equivalent to 100 function evaluations, since one computes $\mathcal{G}(\mathbf{x}_0 + \mathbf{l}_i)$ for each of the 100 EOFs. If we compare the function value reached after a same number of function evaluations, this extra cost is damped for the three first experiments. Indeed, the function value after 900 function evaluations when using the π_i-hierarchy is lower than that using the natural hierarchy after 1000 function evaluations. The difference between the values may even be significant, see the basic method (solid line) versus the basic method with the π_i-hierarchy (dot-dashed line) for the first experiment, or the basic method with $\mathbf{B}$-smoothing (dotted lines) versus the basic method with $\mathbf{B}$-smoothing and the π_i-hierarchy (dashed lines) for all three experiments.

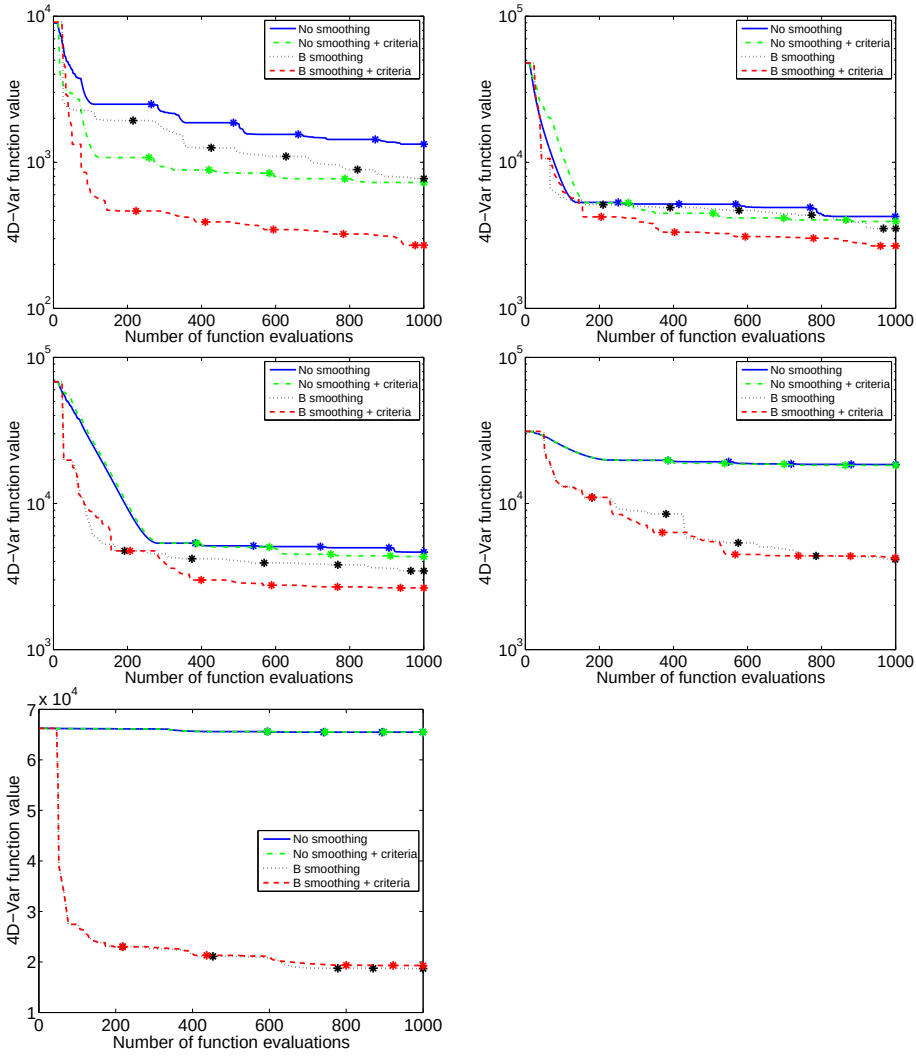

Figure 7.12 — *Convergence of the iterative subspace method. Experiment 1 (top left), experiment 2 (top right), experiment 3 (center left), experiment 4 (center right) and experiment 5 (bottom left).*

For experiments 4 and 5, however, using the EOFs hierarchy based on (7.13) is not profitable since it does not improve the function decrease.

7.11 Numerical comparisons

The goal of this section is twofold. On one side, we compare the numerical performance of our derivative-free ISM method with a EnKF, the existing derivative-free sequential counterpart. On the other side, we illustrate the fact that one cannot expect the performance of derivative-free methods to be similar or outperform methods using derivatives, even at comparable computational cost.

To this aim, we consider three methods. The first is our implementation of the derivative-free ISM Algorithm 7.5 including the **B**-smoothing and the π_i-hierarchy to sort the smoothed EOFs. The second method implements a basic EnKF algorithm with perturbed observations (such as Algorithm 7.3 in Section 7.3.2) and the last one is a classical Gauss-Newton implementation along the lines of Algorithm 6.1 in Section 6.2.

The framework of experimentation is the same as that used in Section 7.7, except that each experiment involves a sequence of data assimilation problems corresponding to a sequence of data assimilation windows, to be as close as possible to operational frameworks (see Table 7.3).

#	n	Length of window	Number of windows
1	816	24 hours	6
2	3168	24 hours	6
3	12480	24 hours	3
4	49536	12 hours	3
5	197376	12 hours	1

Table 7.3 — *Parameters of the 5 different experiments.*

In order to compare the three methods, the same computational budget is allocated. Since the computational cost is dominated by the model integration (direct code), we allow for 500 model integrations per data assimilation problem to solve. For the ISM method, this amounts to 500 function evaluations per window. As in Section 7.7, 100 EOFs are computed, only once at the beginning of the first window in our new framework. These EOFs are then smoothed by **B** and sorted using the EOFs hierarchy based on (7.13) (hence for a cost of 100 function evaluations). These smoothed EOFs and their hierarchy are used for all the windows, restarting from the beginning of the hierarchy for each window. Note that from the second window (if any), no computational budget has to be devoted to the π-hierarchy computation. The basic EnKF algorithm with perturbed observations evolves over the windows with 500 members in the ensemble, according to the budget. For the Gauss-Newton method, assuming at the very worst that the tangent and adjoint codes need 5 times the execution time of the direct code, a budget of 100 CG iterations can be used per window. We have opted for a sequence of 5 incremental 4D-Var subproblems (3.8) at

each window, applying 20 iterations of a preconditioned (by $\mathbf{B}$) CG method for each subproblem.

Figure 7.13 illustrates, for each of the 5 experiments, the *root mean square (RMS) error* between the true state vector and its estimate computed by each of the three methods over the windows. For reference purposes, a free model run (without any data assimilation) is also plotted (dotted lines) on each figure. Comparing first the derivative-free ISM method (solid lines) with the EnKF

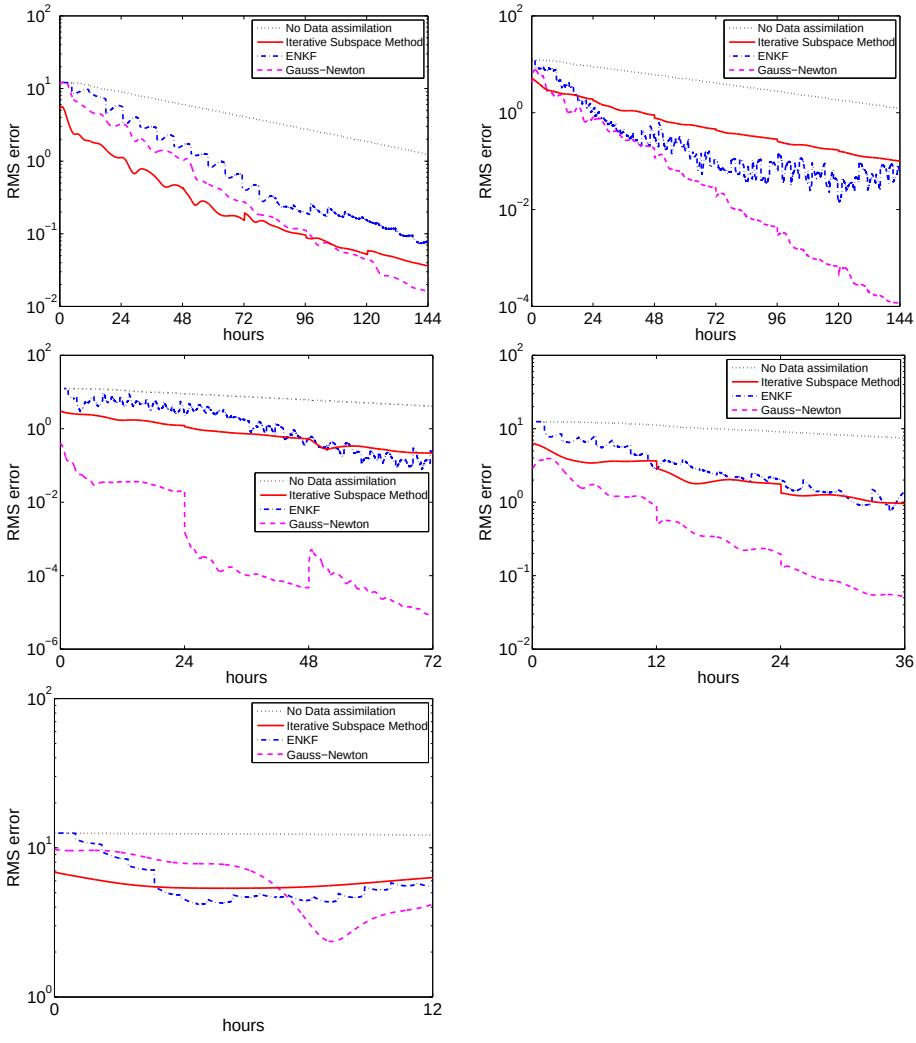

Figure 7.13 — *Comparison with EnKF and Gauss-Newton. Experiment 1 (top left), experiment 2 (top right), experiment 3 (center left), experiment 4 (center right) and experiment 5 (bottom left).*

(dot-dashed lines), roughly speaking, one can say that the quality of their final estimates is comparable for the 5 experiments, meaning that the proposed derivative-free ISM approach is a competitive alternative to the EnKF approach. At comparable computational cost, the final estimates computed by the Gauss-Newton method (dashed lines) clearly outperforms those of the dervative-free methods, corroborating the fact that exploiting the derivatives when possible is a must.

Chapter 8

Numerical experiments with NEMO

In this final chapter, the operational ocean model NEMO is described. Its GYRE configuration is used to illustrate the numerical behavior of the reduced/preconditioned approach from Chapter 6 and of the derivative-free approach from Chapter 7.

8.1 Introduction

NEMO (Nucleus for European Modeling of the Ocean) is a flexible tool for studying the ocean and its interactions with the other components of the earth climate system (Madec, 2008). It is used for operational oceanography seasonal forecast and climate researches. Its components provide numerical solutions of ocean, sea-ice, tracers and biochemistry models. We briefly review its three most important components:

- The OPA (Océan PArallélisé) component models the ocean dynamics and thermodynamics (Madec et al., 1998). It integrates the Navier-Stokes equations on a three spacial dimensional grid using some assumptions made from scale considerations. At each time step, the zonal velocity, the meridional velocity and the sea surface height are computed. Moreover, two active tracers, namely the temperature and the salinity, are also computed using a nonlinear equation of state which couples them to the fluid velocity;

- The LIM (Louvain-la-Neuve sea-ice model) component is used for sea-ice dynamics and thermodynamics (Vancoppenolle et al., 2012). It defines a five-category sea ice thickness, enthalpy, salinity and age distribution

model. The vertical ice growth and decay is determined by an energy conserving thermodynamic model with one layer of snow and five layers of ice. The model also includes a snow ice formation scheme.

- The TOP (Tracer in the Ocean Paradigm) component is used for biogeochemistry (Ethé et al., 2006). It is a passive tracer package including a transport component, and source/sink associated to biogeochemical models.

Besides that, NEMOVAR provides a full data assimilation framework for NEMO with tangent and adjoint codes and the OASIS coupler allows to couple NEMO with several atmospheric general circulation models. The NEMO software is implemented in FORTRAN 90 with preprocessing and runs under UNIX. It is optimized for vector computers and parallelized by domain decomposition with MPI (Message Passing Interface). All inputs and outputs are done using the NetCDF (Network Common Data Format) format. The directory tree of NEMO is illustrated in Figure 8.1 where the three main directories and some subdirectories are represented. The directory `NemoBase` contains the OPA, LIM and TOP components, the directory `NemoVar` contains all the data assimilation framework while the directory `Work` contains input and output files. Overall, there are about 800.000 lines of codes written in 1500 files stored in 200 different directories. The version of NEMO used for the numerical experiments is 9.0 (2005) and the version of Nemovar is 0.2 (2007).

There are 6 available configurations at the moment. For our numerical experiments, the GYRE configuration is used. This configuration, that we shortly describe for completeness, consists in an idealized double gyre that mimics the North Atlantic circulation with its Gulf stream current. The basin from 15°N to 50°N and from 85°W to 55°W is rotated with an angle of 45 degrees in order to capture the maximum length of the Gulf stream current (see Figure 8.2). The basin has 3180 km length, 2120 km width and $4,5$ km depth. The horizontal discretization grid is uniform with 32 points in the length direction and 22 points in the width direction. The vertical discretization is nonuniform with 31 vertical levels whose thickness varies from approximately 5 meters at the surface to 250 meters near the bottom. There are thus $32 \times 22 \times 31 = 21824$ grid points. They are represented in Figure 8.3 where the grid points at the surface and at the bottom are plotted in blue and red, respectively. The ocean model in its GYRE configuration is forced analytically by a penetrative solar radiation with a Newtonian type net heat flux, with a fresh water flux and with a wind stress which varies zonally and seasonally. This model computes the sea level anomaly at the surface grid points and the velocities (zonal and meridional) at each grid point. The evolution of two tracers, namely the temperature and the salinity, is also computed at each grid point. Thus, the size of the state vector is fixed to $(1 \times 32 \times 22) + (4 \times 32 \times 22 \times 31) = 88000$. To obtain an equilibrium where the main features of the configuration (gyres and currents) are stable, the model is spun-up from rest

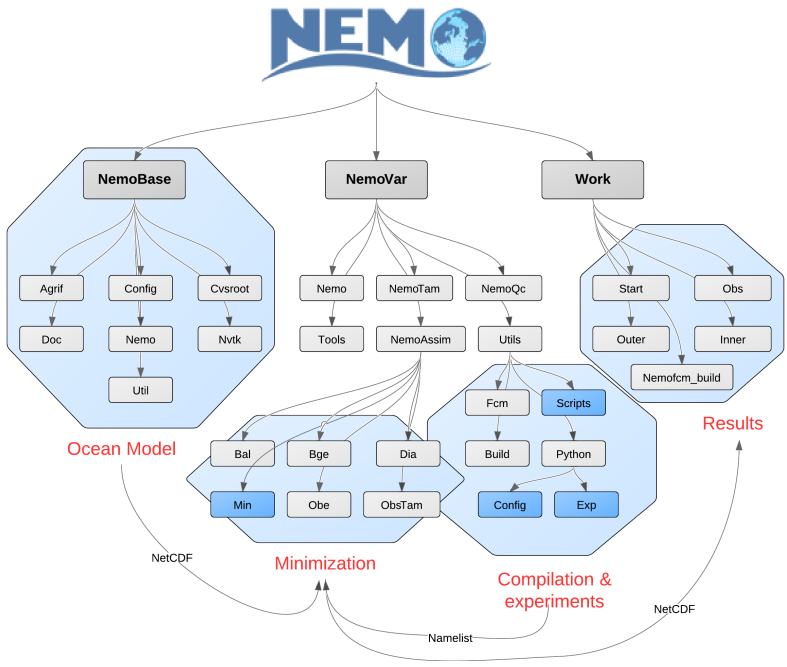

Figure 8.1 — *Directory tree for NEMO (illustrated with Lucyd-Chart™).*

for 100 years (thereafter counted as years 1 to 100). The four outputs of the model at the sea surface for the year 100 are illustrated in Figure 8.4. From the two first pictures, one can remark that the model simulates a stream function with a large anticyclonic subtropical gyre and a weaker cyclonic subpolar gyre.

The length of the data assimilation window used for our numerical experiments begins after the spin-up of 100 years and is fixed to 6 days. The time step for the model integration is fixed to 5760 seconds, so that the number of observation times considered amounts to 90. For the whole data assimilation problem, there are 1946 measurements of temperature, 1297 measurements of salinity and 2240 measurements of sea level anomaly, distributed in space and time. No velocity observations are available. The runs have been performed on an Intel Core 2 Duo (Dual Core 2.4 GHz) with 4096 KB of cache memory and 2 GB of RAM memory. The CPU time to run the direct, the tangent and the adjoint codes for 6 days is summarized in Table 8.1. The CPU time for a product by the background error covariance matrix $\mathbf{B}$ is also given.

In Chapter 6, the Ritz-Galerkin starting point and the limited memory preconditioner (LMP) have been presented as an attempt to accelerate further the Gauss-Newton method further. Their behaviors have been numerically illustrated on a shallow water model in Section 6.5 and their positive impacts have

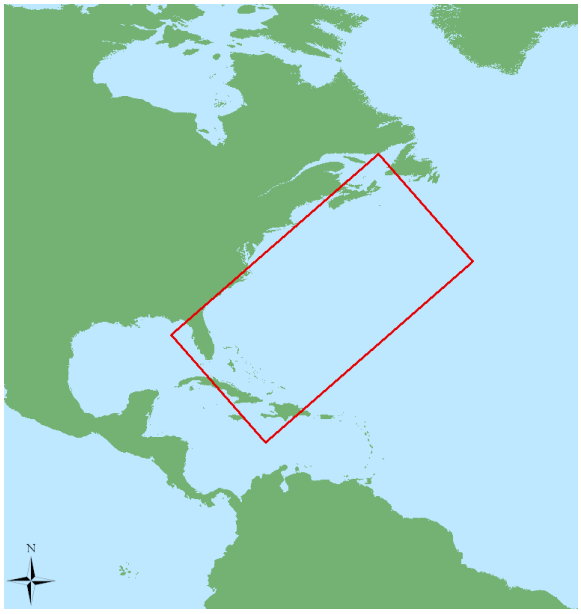

Figure 8.2 — *GYRE domain (illustrated with ArcGIS™).*

Code	CPU time
Direct	2.3 sec
Tangent	1.6 sec
Adjoint	2.2 sec
Product by **B**	0.04 sec

Table 8.1 — *CPU times for the direct, the tangent and the adjoint codes and for the product by* **B**.

been exposed. In the next section, this reduced and preconditioned approaches are illustrated using the NEMO framework.

8.2 Reduced and preconditioned approaches

The reduced and preconditioned approaches are based on the use of information contained in the EOFs. The computation of such directions have been described in Subsection 4.3.3 and relies on a set of vectors which are representative of the variability of the system. In the NEMO framework, this set is constructed using one state vector for each day of the 60th years ($m = 365$). The criteria defined in (4.31) which estimates the percentage of variation accounted for by the first r EOFs is shown in Figure 8.5. We see that retaining the first five

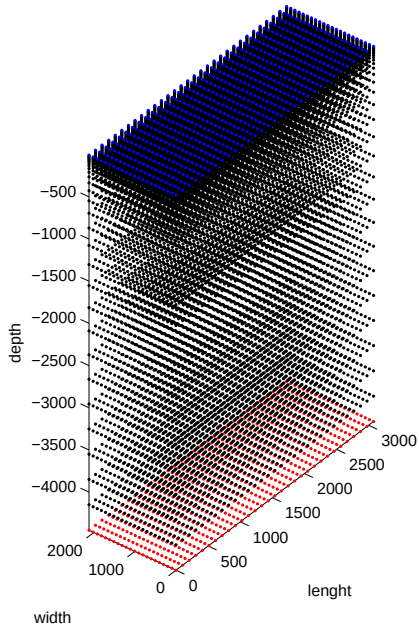

Figure 8.3 — *discretization grid.*

EOFs enables to explain 82% of the system's variability. As for the numerical experiments on the shallow water model presented in Section 6.5, the size of the subspace is fixed to $r = 5$. This choice is a good compromise since it allows to account for most of the variation without increasing too much the dimension of the subspace spanned by the selected EOFs.

We illustrate the impact of the Ritz-Galerkin starting point (6.28) and of the LMP (6.31) when the subspace spanned by $\mathbf{L_0}$ is defined by the five first dominant EOFs. The same experimental framework is considered by allowing three outer Gauss-Newton iterations with five inner CG iterations to solve each linear system. Figures 8.6 shows the history of the nonlinear and incremental cost functions for the three outer iterations (the nonlinear with markers and the incremental with lines). Note that the nonlinear function value is available only at outer iterations when a linearization is performed in the Gauss-Newton algorithm. For the incremental cost functions, the curves are placed one after the other in sequence and the inner iterations are cumulated. The dash-dot curve is obtained using a basic Gauss-Newton algorithm with the background covariance matrix $\mathbf{B}$ as preconditioner in the CG method. The dotted curve and the dashed curve are obtained using the same approach, but with the Ritz-Galerkin starting point (6.28) for the first system. No second-level preconditioner has been added to generate the dotted curve, while the LMP (6.31) has been used as second-level preconditioner to get the dashed curve. A first

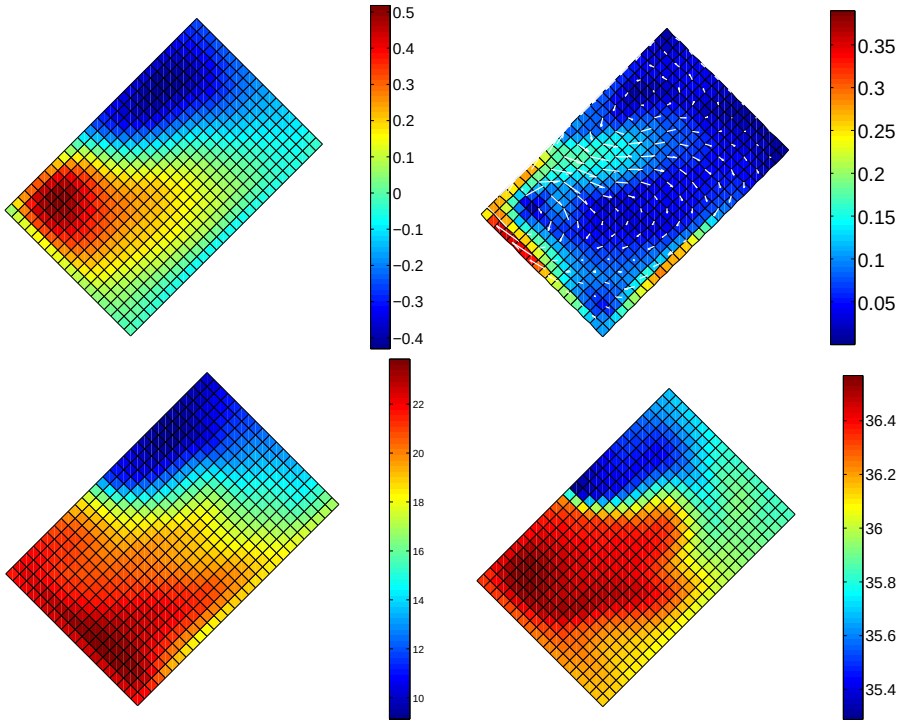

Figure 8.4 — *Outputs of NEMO in its GYRE configuration at the sea surface for the year* 100. *Sea level anomaly in meter (top left), velocities in meter per second (top right), temperature in Celsius degree (bottom left) and salinity (bottom right).*

look at this picture shows that the 4D-Var function is much more nonlinear that the one from the shallow water model (compare with Figure 6.3, for example). The nonlinear gaps make the optimization more hazardous since a decrease in the quadratic function is less related to a decrease in the nonlinear 4D-var function. Remembering that the computation of the Ritz-Galerkin starting point requires five matrix-vector products and that the LMP is available without any extra cost, it makes sense to compare the results obtained after 15 cumulated inner iterations of the basic approach (last circle) with that obtained after 10 cumulated inner iterations of the two others (next to last dot and square). One can see that the Ritz-Galerkin approach (with or without the LMP) does not make an improvement for the same computational effort (15 matrix-vector products).

We have addressed this failure by performing an extra linearization after the computation of the Ritz-Galerkin starting point. The results are presented in Figure 8.7. Note that now the first quadratic function minimized for the

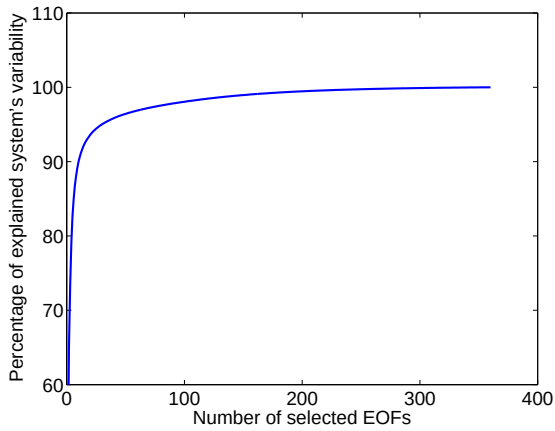

Figure 8.5 — *Percentage of explained system's variability.*

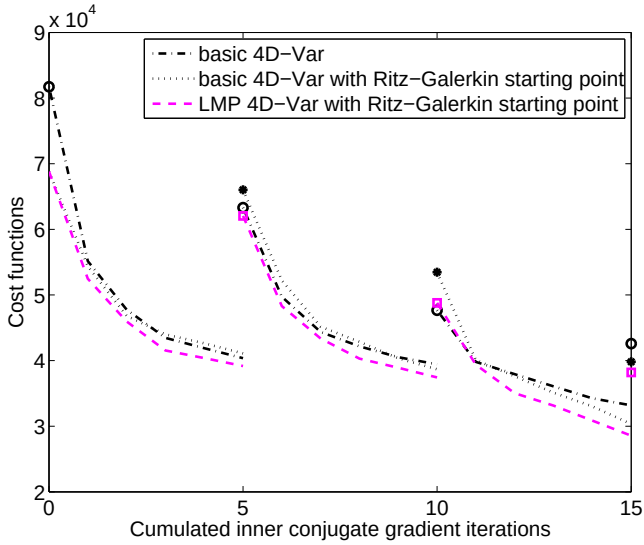

Figure 8.6 — *Convergence curves with the NEMO framework.*

Ritz-Galerkin approach (with or without the LMP) is not the same that the one for the basic 4D-Var because an extra linearization has been performed. With this extra linearization, the Ritz-Galerkin starting point improves the method as the next to last dot is lower than the last circle. Moreover, the free-available LMP gives a stronger improvement as the next to last square is lower than the next to last dot.

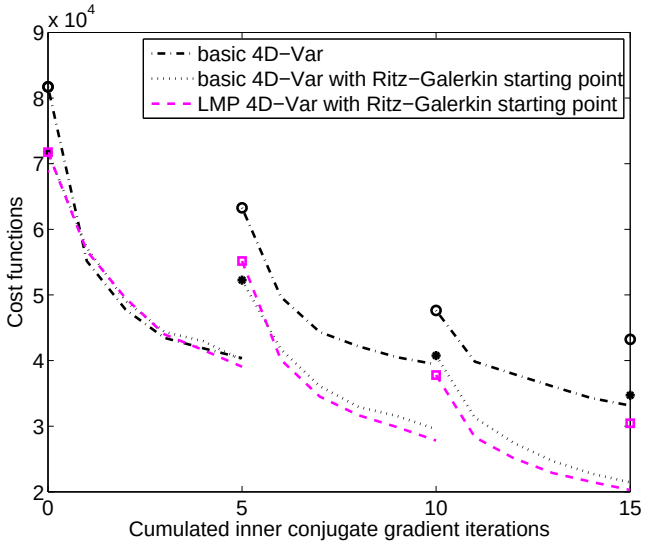

Figure 8.7 — *Convergence curves with an extra linearization after the Ritz-Galerkin starting point.*

8.3 Derivative-free approach

In Chapter 7, an attempt to solve the 4D-Var problem using a derivative-free approach has been presented. This method is based on the construction of a sequence of well-chosen and low dimensional subspaces and exploits reduction techniques using EOFs. The numerical results presented in Section 7.10 and 7.11 show that it is a promising approach. In this section, we illustrate the behavior of the proposed approach in the NEMO framework by beginning with a short study about the quality of the EOFs.

The oscillatory behavior of the EOFs has been analyzed in Section 7.8 on the two-dimensional shallow water model. We have illustrated that the EOFs corresponding to the largest singular values are smoother than the EOFs corresponding to the smallest singular values which are generally oscillating. The same conclusion can be drawn for NEMO and is illustrated in Figure 8.8, where the components of the sea surface temperature for the first and the 10th EOFs have been plotted. While the first EOF is very smooth, one observes an oscillatory and discontinuous behavior already in the 10th EOF. As in the shallow water case, a smoothing based on the background error covariance matrix $\mathbf{B}$ reduces effectively this inconvenient behavior (see Figure 8.9). Once the smoothed EOFs have been calculated, it is possible to compute the π_i-hierarchy based on (7.13), for $i = 1, \ldots, 365$ that induces a new order in the EOFs. The criterion values (7.13) are presented in Figure 8.10.

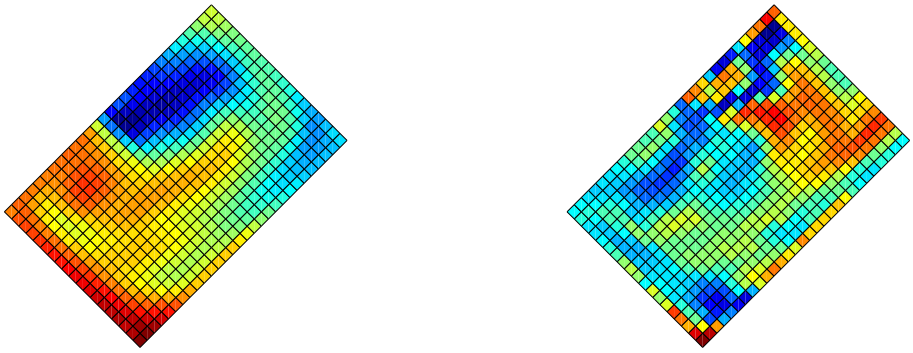

Figure 8.8 — *EOF number* 1 *(left) and EOF number* 10 *(right)* .

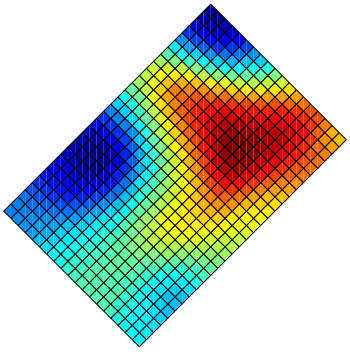

Figure 8.9 — *EOF number* 10 *smoothed by* **B**.

The main result of this section is presented in Figure 8.11, where the four different strategies (basic method, basic method with **B**-smoothing, basic method with the π_i-hierarchy and basic method with **B**-smoothing and the π_i-hierarchy) of the ISM method are compared. For this experiment, we consider the same experimental framework as in Sections 7.7 and 7.10, using the same values for the parameters, except that the limit on the number of function evaluations has now been increased to 2000. One clearly sees that each modification improves the performance individually, while the cumulated effect of using the **B**-smoothing and the π_i-hierarchy brings a further improvement and avoids a stagnation in the decrease process of the function. Considering now the cost of implementing each modification, the application of **B** to an EOF is cheap compared to a function evaluation (see Table 8.1), so that smoothing by **B** can be considered as a negligible operation. The cost of using the EOFs hierarchy based on (7.13), however, is equivalent to 365 function evaluations. But if we compare the function value reached after a same number of function

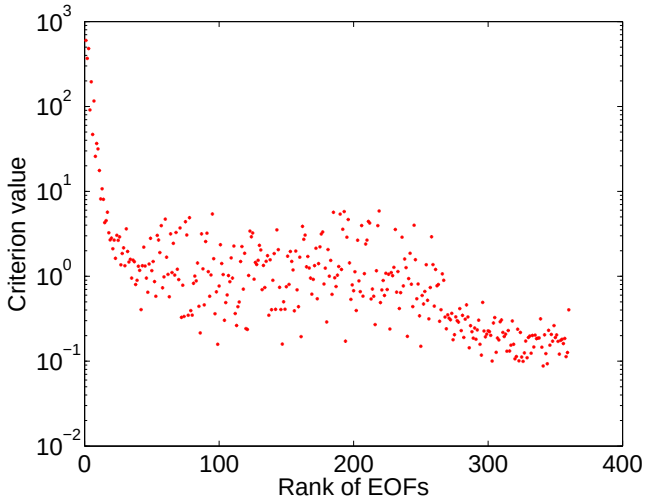

Figure 8.10 — *Criterion value for each EOF.*

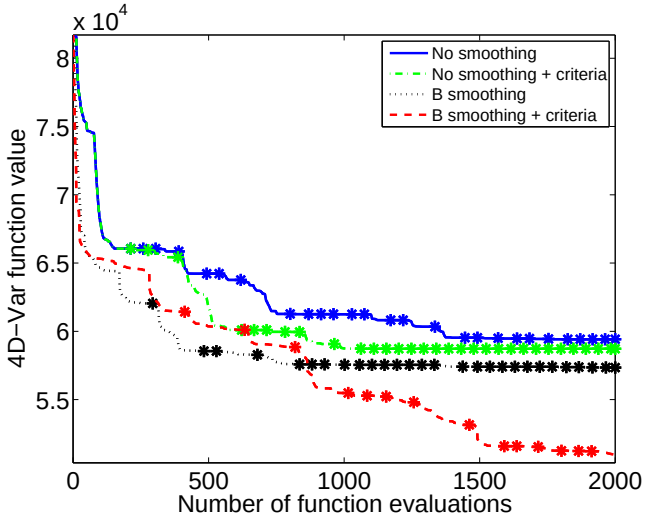

Figure 8.11 — *Results of the ISM method for the four strategies.*

evaluations, this extra cost is damped (see Table 8.2). Indeed, the function value after 1635 function evaluations when using the π_i-hierarchy is lower than that using the natural hierarchy after 2000 function evaluations for both cases (with and without **B**-smoothing). The difference between the function values may even be significant, see on Figure 8.11 the basic method with **B**-smoothing (dotted lines) versus the basic method with **B**-smoothing and the π_i-hierarchy

(dashed lines).

Method	Function value
No smoothing	59420
No moothing + π_i-hierarchy	58720
B-smoothing	57340
B-smoothing + π_i-hierarchy	51590

Table 8.2 — *The function value after a budget of* 2000 *function evaluations.*

To our knowledge, no EnKF has been implemented and validated in the NEMO framework. The development of such a filter in a real operational model is not straightforward and its validation is a difficult task requiring a twin experiment framework. For these reasons, the comparison between the ISM method and a basic EnKF, such the one presented on a shallow water model in Section 7.11, is not possible on the NEMO framework at the moment. However, we can compare the ISM method with the classical Gauss-Newton method available in the data assimilation framework of NEMO. As in the shallow water model, we illustrate the fact that the derivative-free approach cannot outperform the Gauss-Newton method, even at a comparable computational cost. To draw this conclusion, we first analyze the computational cost of the Gauss-Newton method in terms of 4D-Var function evaluations. At each outer loop, a model integration is performed to evaluate the 4D-Var function and to build the linearization (see Algorithm 6.1). This cost is equivalent to one 4D-Var function evaluation. Moreover, at each inner loop, the derivative of the incremental 4D-Var is computed (to be used in the CG-like method), requiring one call to the tangent code and one to the adjoint code. Assuming from Table 8.1 that one inner iteration costs two 4D-Var function evaluations (since the tangent and adjoint codes require roughly the same computational cost than the direct code), one can then consider that the cost of one outer iteration amounts to twice the number of inner iterations plus one. The results of the Gauss-Newton method are presented in Figure 8.12 when three outer loops with five inner loops each are performed (thus with an associated cost of 11 4D-Var function evaluations per outer iteration). The 4D-Var function is evaluated four times (the four circles), giving three function value decreases. The computational cost to obtain each of these decreases in terms of 4D-Var function evaluations (using the above assumption) are summarized in Table 8.3. The corresponding decreases in the function are compared with those obtained for the best variant of the ISM method (with **B**-smoothing and the π_i-hierarchy) at the same computational cost (using the results from Figure 8.11). The results of this table are represented in Figure 8.13. It clearly shows, for the same computational cost, the low performance of the ISM method compared to the Gauss-Newton method. If we now extend the computational budget

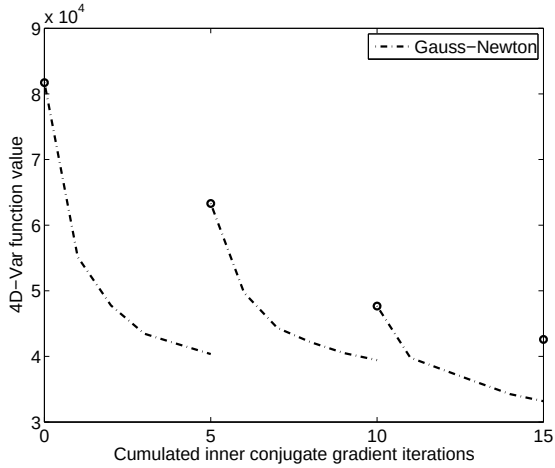

Figure 8.12 — *Convergence of the Gauss-Newton method.*

Computational cost	Function value	
	Gauss-Newton	ISM
1	81720	81720
12	63370	72570
23	47670	68750
34	42580	66340

Table 8.3 — *Computational cost and function value.*

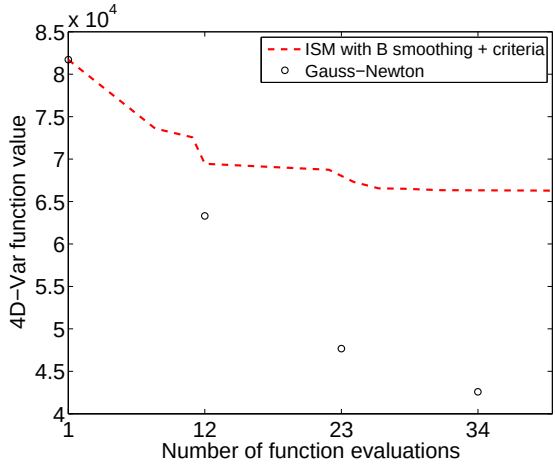

Figure 8.13 — *Computational cost and function value.*

for the ISM method, we can observe (from Figure 8.11) that it needs 280 function evaluations to reach the same decrease than the one obtained after the first outer iteration of Gauss-Newton (see Table 8.3). It means that the ISM method needs more than 20 times the number of 4D-Var function evaluations used to obtain the same function decrease. Moreover, with a budget of 2000 function evaluations, the ISM method does not even reach the decrease obtained after the second outer iteration of the Gauss-Newton method.

Conclusions and Perspectives

Marine and weather forecasts are used daily to make economic decisions mainly in agriculture, energy and transportation industries. In the last twenty years, major improvements have been performed to collect more observations with radars and satellites, to model accurately the earth system and to combine these information properly for producing forecast. By doing so, a five-day weather forecast of today is as reliable as a two-day weather forecast 20 years ago.

The research work described in this thesis was concerned with the development and the study of new optimization methods for solving data assimilation problems with application in oceanography. More particularly, we have developed a reduced-preconditioned and a derivative-free approach for solving the 4D-Var problem. In the next two paragraphs, both kinds of developments are reviewed and the results are summarized. At the same time, we also suggest some possible directions for future research works.

A reduced and preconditionned approach

Motivated by the theoretical equivalence between the SEEK filter and a reduced variant of the 4D-Var problem, we have investigated the use of information contained in the dominant Empirical Orthogonal Functions (EOFs) to further accelerate the Gauss-Newton method used to minimize the 4D-Var function. We have first derived an appropriate starting point for the conjugate gradient-like (CG) method which is used to solve the first linear system in the sequence generated by the Gauss-Newton method. This starting point is the solution of a reduced incremental 4D-Var problem and is equivalent to the Ritz-Galerkin starting point with respect to the selected EOFs. In order to further exploit the EOFs information computed to obtain the Ritz-Galerkin starting point, we have then proposed a "second-level preconditioner" from the Limited-Memory Preconditioner (LMP) class introduced by Gratton et al. (2011). The combination of these two improvements have been numerically assessed using, firstly, an academical shallow water model and, secondly, the GYRE configuration of

the NEMO framework. We have shown, in both cases, that the combination accelerates the convergence of the incremental method.

However, a larger set of data assimilation problems using an operational configuration over the whole ocean is needed to perform a rigorous quantitative study. Further research could investigate the use of informations (such as Ritz pairs or descent directions) gained when solving systems by the CG method to enrich the LMP based on the EOFs for the next systems in the sequence generated by the Gauss-Newton method.

A derivative-free approach

Evaluating derivatives in operational data assimilation problems is challenging as one needs to compute the Jacobian of the model operator and its transpose, which implies the derivation of a tangent linear and adjoint code. The Ensemble Kalman Filter (EnKF) provides a suitable derivative-free alternative by using a Monte-Carlo implementation on the Kalman filter equations. However, no derivative-free variant of the variational approach has been proposed so far (to the knowledge of the author of this research). Our derivative-free Iterative Subspace Method (ISM) minimizes the 4D-Var function by building a sequence of subspaces from information contained in the EOFs and exploring each of them using a coordinate-search method. Two strategies to improve the quality of the subspaces have been presented. The first one is based on smoothing techniques applied to the EOFs while the second gives a new EOFs hierarchy for the construction of the subspaces. Numerical tests on an academical shallow water model have shown that the ISM method is a competitive alternative to a basic EnKF method since it produces final estimates with comparable quality at similar cost. This conclusion could not be extended using the GYRE configuration of the NEMO framework due to the lack of an EnKF implementation in this framework. Nevertheless, a comparison with the Gauss-Newton method (using derivatives) shows that the ISM method needs more than 20 times more function evaluations that the Gauss-Newton method to obtain the same function decrease. It corroborates the fact that exploiting the derivatives when possible is imperative.

Since it is a first derivative-free attempt to solve the 4D-Var problem, different research perspectives are possible. As previously, it could be interesting to assess this method on a larger set of data assimilation problems with the aim of reinforcing the numerical experimentations. Although one cannot expect the performance of derivative-free methods to be comparable to that of methods based on derivatives, future research works may be investigated to narrow this gap. The derivative-free solver SID-PSM used for the subspace minimization in Chapter 7 depends on multiple parameters. It could be interesting to perform a study on their optimal values and to see if it is possible to tune some of them. It would also be worth investigating whether another derivative-free algorithm

which takes the underlying nature of the problem into account could be more suitable.

Finally, it is our hope that the work described in this thesis will be useful in future algorithmic developments for solving data assimilation problems.

Contributions

Our contribution is the study and the design of new efficient optimization methods for solving data assimilation problems. These numerical methods draw their theoretical foundations from the connection between the SEEK filter and its reduction technique. We summarize our contributions below:

- The theoretical connection between the SEEK filter and a reduced variant of the 4D-Var problem has been proven;
- An appropriate starting point with a powerful limited memory preconditioner has been derived for the incremental approach. They have been successfully implemented in the operational ocean model NEMO where numerical simulations have been performed using the GYRE configuration.
- Smoothing techniques and selection criteria have been developed to build a sequence of appropriate low dimensional subspaces which are used by a derivative-free solver for minimizing the 4D-Var function. This approach have been successfully implemented in the operational ocean model NEMO where numerical simulations have been performed using the GYRE configuration.

These contributions are the subjects of two papers:

- S. Gratton, P. Laloyaux, A. Sartenaer, and J. Tshimanga. A reduced and limited-memory preconditioned approach for the 4D-var data assimilation problem. *Quarterly Journal of the Royal Meteorological Society*, 137:452-466, 2011a.
- S. Gratton, P. Laloyaux and A. Sartenaer. Derivative-free optimization for large-scale nonlinear data assimilation problems. *Quarterly Journal of the Royal Meteorological Society*, In preparation.

Main notations and abbreviations

Matrices, vectors and operators

m	Number of vectors in an EOF analysis	48
N	Number of observation times in a data assimilation window	14
n	Size of the state vector	7
p	Size of the observation vector $\mathbf{y}$	14
p_i	Size of the observation vector $\mathbf{y}_i$	10
t_0	Initial time of a data assimilation window	14
t_N	Final time of a data assimilation window	14

Space dimension and constant parameters

$\mathbf{B}$	Background error covariance matrix ($\mathbb{R}^{n\times n}$)	15
$\boldsymbol{\epsilon}_i^o$	Error in the observation $\mathbf{y}_i$ ($\mathbb{R}^{p_i}$)	10
$\boldsymbol{\epsilon}_i^x$	error in the true state vector $\mathbf{x}_i^t$ ($\mathbb{R}^n$)	31
$\mathbf{H}_i$	Linear observation operator at time t_i ($\mathbb{R}^{p_i\times n}$)	10
$\mathbf{H}_i^k$	Jacobian matrix of $\mathcal{H}_i$ at x_k ($\mathbb{R}^{p_i\times n}$)	20
$\mathbf{K}_i$	Kalman gain ($\mathbb{R}^{n\times p_i}$)	37
$\mathbf{L}$	Basis of a reduced space ($\mathbb{R}^{n\times r}$)	26
$\mathbf{L}_0$	Basis of a reduced space built with eigenvectors ($\mathbb{R}^{n\times r}$)	44
$\mathbf{M}_{i,0}^k$	Jacobian matrix of $\mathcal{M}_{i,0}$ at $\mathbf{x}_k$ ($\mathbb{R}^{n\times n}$)	23
$\mathbf{M}_{i+1,i}$	Linear model operator from time t_i to time t_{i+1} ($\mathbb{R}^{n\times n}$)	8
$\mathbf{R}$	Error covariance matrix of $\mathbf{y}$ ($\mathbb{R}^{p\times p}$)	22
$\mathbf{R}_i$	Error covariance matrix of $\mathbf{y}_i$ ($\mathbb{R}^{p_i\times p_i}$)	13
$\mathbf{S}$	Sample covariance matrix ($\mathbb{R}^{n\times n}$)	49
$\mathbf{x}^a$	Analysis vector ($\mathbb{R}^n$)	18

List of Algorithms

Bibliography

B. Anderson and J. Moore. *Optimal Filtering*. Prentice-Hall, 1979.

E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. SIAM, 1999.

A. Andrews. A square root formulation of the Kalman covariance equations. *AIAA Journal*, 6:1165–1166, 1968.

A. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM, 2005.

D. Ashlock. *Evolutionary Computation for Modeling and Optimization*. Springer, 2005.

R. Aster, B. Borchers, and C. Thurber. *Parameter Estimation and Inverse Problem*. Elsevier, 2005.

Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, 1987.

R. Battin. *Astronautical Guidance*. McGraw-Hill, 1964.

B. Bell and F. Cathey. The iterated Kalman filter update as a Gauss-Newton method. *IEEE Transactions on Automatic Control*, 38:294–297, 1993.

J. Bellantoni and K. Dodge. A square root formulation of the Kalman-Schimdt filter. *AIAA Journal*, 5:1309–1314, 1968.

A. Bennett. *Inverse modeling of the ocean and atmosphere*. Cambridge University Press, 2002.

M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182:418–477, 2002.

L. Biegler, G. Biros, O. Ghattas, M. Heinkenschloss, D. Keyes, B. Mallick, Y. Marzouk, L. Tenorio, B. van Bloemen Wanders, and K. Willcox. *Large-Scale Inverse Problems and Quantification of Uncertainty*. Wiley, 2011.

G. Bierman. *Factorization Methods for Discrete Sequential Estimation*. Academic Press, 1977.

V. Bjerknes. Das problem der wettervorhersage, betrachtet vom standpunkte der mechanik und der physik. *Meteorologische Zeitschrift*, 21:1–7, 1904.

A. Bjorck. *Numerical Methods for Least Squares Problems*. SIAM, 1996.

L. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. Whaley. *ScaLAPACK Users' Guide*. SIAM, 1997.

M. Bonavita and E. Holm L. Isaksen. Use of eda-based background error variance in 4D-Var. *ECMWF Newsletter*, 130, 2012.

M. Bonavita, L. Isaksen, and E. Holm. On the use of EDA background error variances in the ECMWF 4D-Var. *Quarterly Journal of the Royal Meteorological Society*, 2012.

A. Booker, J. Dennis, P. Frank, D. Serafini, V. Torczon, and M. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17:1–13, 1998.

K. Borre and G. Strang. *Linear Algebra, Geodesy, and Gps*. Wellesley Cambridge Press, 1997.

M. Brand. Fast online SVD revisions for lightweight recommender systems. *Proceedings of SIAM International Conference on Data Mining*, pages 37–47, 2003.

P. Brasseur and J. Verron. The SEEK filter method for data assimilation in oceanography: a synthesis. *Ocean Dynamics*, 56:650–661, 2006.

R. Byrd, R. Schnabel, and G. Shultz. Parallel quasi-newton methods for unconstrained optimization. *Mathematical Programming*, 42:273–306, 1988.

N. Carlson. Fast triangular formulation of the square root filter. *AIAA Journal*, 11:1259–1265, 1973.

K. Chen. *Matrix Preconditioning Techniques and Applications*. Cambridge University Press, 2005.

M. Chung. Heat kernel smoothing on unit sphere. *Proceedings of IEEE International Symposium on Biomedical Imaging*, pages 992–995, 2006.

B. Cipra. Engineers look to Kalman filtering for guidance. *SIAM News*, 26, 1993.

N. Clark, L. Eber, R. Laurs, J. Renner, and J. Saur. Heat exchange between ocean and atmosphere in the eastern north pacific for 1961-1971. Technical report, National Oceanic and Atmospheric Administration, 1974.

A. Conn, N. Gould, and Ph. Toint. *Trust-Region Methods*. SIAM, 2000.

A. Conn, K. Scheinberg, and L. Vicente. *Derivative-Free Optimization*. SIAM, 2009.

P. Courtier. Dual formulation of four-dimensional variational assimilation. *Quarterly Journal of the Royal Meteorological Society*, 123:2449–2461, 1997.

P. Courtier and O. Talagrand. Variational assimilation of meteorological observations with the adjoint vorticity equation. II: Numerical results. *Quarterly Journal of the Royal Meteorological Society*, 113:1329–1347, 1987.

P. Courtier, J. Thépaut, and A. Hollingsworth. A strategy for operational implementation of 4D-Var, using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, 120:1367–1388, 1994.

B. Cushman-Roisin and J.-M. Beckers. *Introduction to Geophysical Fluid Dynamics: Physical and Numerical Aspects*. Academic Press Inc, 2010.

A. Custodio and L. Vicente. Using sampling and simplex derivatives in pattern search methods. *SIAM Journal on Optimization*, 18:537–555, 2007.

A. Custodio, H. Rocha, and L. Vicente. Incorporating minimum Frobenius norm models in direct search. *Computational Optimization and Applications*, 46:265–278, 2010.

N. Daget, A. Weaver, and M. Balmasedab. Ensemble estimation of background-error variances in a three-dimensional variational data assimilation system for the global ocean. *Quarterly Journal of the Royal Meteorological Society*, 135:1071–1094, 2009.

T. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, 2006.

J. E. Dennis and Virginia Torczon. Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1:448–474, 1991.

S. Durbiano. *Vecteurs caractéristiques de modèles océaniques pour la réduction d'ordre en assimilation de données*. PhD thesis, Université Joseph Fourier, 2001.

M. Eaton. *Multivariate Statistics: a Vector Space Approach*. John Wiley and Sons, 1983.

C. Ethé, O. Aumont, M-A Foujols, and M. Lévy. Nemo reference manual, tracer component: Nemo-top. *Note du Pole de Modélisation, Institut Pierre-Simon Laplace*, 2006.

G. Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics. *Journal of Geophysical Research*, 99:10143–10162, 1994.

G. Evensen. The Ensemble Kalman filter: theoretical formulation and practical implementation. *Ocean Dynamics*, 53:343–367, 2003.

G. Evensen. Sampling strategies and square root analysis schemes for the EnKF. *Ocean Dynamics*, 54:539–560, 2004.

G. Evensen. *Data Assimilation: The Ensemble Kalman Filter*. Springer, 2007.

R. Giering and T. Kaminski. Recipes for adjoint code construction. *ACM Transactions on Mathematical Software*, 24:437–474, 1998.

G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, third edition, 1996.

S. Gratton, A. Lawless, and N. Nichols. Approximate gauss-newton methods for nonlinear least squares problems. *SIAM Journal on Optimization*, 18: 106–132, 2007.

S. Gratton, A. Sartenaer, and J. Tshimanga. On a class of limited memory preconditioners for large scale linear systems with multiple right-hand sides. *SIAM Journal on Optimization*, 21:912–935, 2011.

M. Grewal, L. Weill, and A. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration*. Wiley Press, 2000.

A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, 2008.

I. Griva, S. Nash, and A. Sofer. *Linear and Nonlinear Optimization*. SIAM, 2009.

J. Hahn and T. Edgar. An improved method for nonlinear model reduction using balancing of empirical grammians. *Computers and Chemical Engineering*, 26:1379–1397, 2002.

A. Hannachi, I. Jolliffe, and D. Stephenson. Empirical orthogonal functions and related techniques in atmospheric science: A review. *International Journal of Climatology*, 27:1119–1152, 2007.

M. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

J. Horel. A rotated principal component analysis of the interannual variability of the northern hemisphere 500 mb height field. *Monthly Weather Review*, 109: 2080–2092, 1981.

I. Hoteit and D. T. Pham. Evolutivity of the reduced state space and data assimilation schemes based on the Kalman filter. *Journal of the Meteorological Society of Japan*, 81:21–39, 2003.

B. Hunt, E. Kostelich, and E. Szunyogh. Efficient data assimilation for spatiotemporal chaos: A local Ensemble transform Kalman filter. *Physica D*, 230:112–126, 2007.

K. Ide, P. Courtier, M. Ghil, and A.C. Lorenc. Unified notation for data assimilation: operational, sequential and variational. *Journal of the Meteorological Society of Japan*, 75:181–189, 1997.

I. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics, 2002.

I. Jolliffe, N. Trendafilov, and M. Uddin. A modified principal component technique based on the LASSO. *Journal of Computational and Graphical Statistics*, 12:531–547, 2003.

S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. *Proceedings of the IEEE*, pages 182–193, 1997.

S. Julier and J. Uhlmann. Unscented filtering and nonlinear estimation. In *Proceedings of the IEEE*, pages 401–422, 2004.

R. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82:35–45, 1960.

E. Kalnay. *Atmospheric modeling, data assimilation and predictability.* Cambridge University Press, 2003.

E. Kalnay, H. Li, T. Miyoshi, S.-C. Yang, and J. Ballabrera-Poy. 4D-Var or Ensemble Kalman filter? *Tellus A*, 59:758–773, 2007.

P. Kaminski. Discrete square root filtering: A survey of current techniques. *IEEE Transactions on Automatic Control*, 16:727–736, 1971.

R. Katz and A. Murphy. *Economic Value of Weather and Climate Forecasts.* Cambridge University Press, 1997.

M. Krysta, E. Blayo, E. Cosme, and J. Verron. A consistent hybrid variational-smoothing data assimilation method: Application to a simple shallow water model of the turbulent mid-latitude ocean. *Monthly Weather Review*, 139: 3333–3347, 2011.

S. Lall, J. Marsden, and S. Glavaski. Empirical model reduction of controlled nonlinear systems. *Proceedings of the IFAC World Congress*, pages 473–478, 1999.

S. Laroche and P. Gauthier. A validation of the incremental formulation of 4D variational data assimilation in a nonlinear barotropic flow. *Tellus A*, 50: 557–572, 1998.

A. Lawless, N. Nichols, and S. Ballard. A comparison of two methods for developing the linearization of the shallow water model. *Quarterly Journal of the Royal Meteorological Society*, 129:1237–1254, 2003.

A. Lawless, S. Gratton, and N. Nichols. Approximate iterative method for variational data assimilation. *International Journal for Numerical Methods in Fluids*, pages 1–6, 2004.

A. Lawless, N. Nichols, C. Boess, and A. Bunse-gerstner. Using model reduction methods within incremental 4D-Var. Technical report, The university of Reading, 2006.

F.-X. Le Dimet and O. Talagrand. Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. *Tellus*, 38A: 97–110, 1986.

J. Lewis and S. Lakshmivarahan. Sasaki's pivotal contribution: Calculus of variations applied to weather map analysis. *Monthly Weather Review*, 136: 3553–3567, 2008.

J. Lewis, S. Lakshmivarahan, and S. Dhall. *Dynamic Data Assimilation: A Least Squares Approach*. Cambridge University Press, 2006.

R. Lewis, V. Torczon, and M. Trosset. Direct search methods: Then and now. *Journal of Computational and Applied Mathematics*, 124:191–207, 2000.

Z. Li and I. M. Navon. Optimality of 4D-Var and its relationship with the Kalman filter and Kalman smoother. *Quarterly Journal of the Royal Meteorological Society*, 127:661–684, 2001.

C. Lieberman, K. Willcox, and O. Ghattas. Parameter and state model reduction for large-scale statistical inverse problem. *SIAM Journal on Optimization*, 32:2523–2542, 2010.

J. Lions, P. Manley, R. Temam, and S. Wang. Physical interpretation of the attractor dimension for the primitive equations of atmospheric. *Journal of the Atmospheric Sciences*, 54:1137–1143, 1997.

A. Lorenc. Modelling of error covariances by 4D-Var data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 129:3167–3182, 2003.

A. Lorenc and F. Rawlins. Why does 4D-Var beat 3D-Var? *Quarterly Journal of the Royal Meteorological Society*, 131:3247–3257, 2005.

P. Lynch. The origins of computer weather prediction and climate modelling. *Journal of Computational Physics*, 227:3431–3444, 2008.

L. Rocha M. Wall, A. Rechtsteiner. Singular value decomposition and principal component analysis. *A practical Approach to Microarray Data Analysis*, pages 91–109, 2003.

G. Madec. Nemo ocean engine. *Note du Pole de Modélisation, Institut Pierre-Simon Laplace*, 2008.

G. Madec, P. Delecluse, M. Imbard, and C. Lévy. Opa 8.1 ocean general circulation model reference manual. *Note du Pole de Modélisation, Institut Pierre-Simon Laplace*, 1998.

J.-F. Mahfouf and F. Rabier. The ECMWF operational implementation of four-dimensional variational assimilation. II: Experimental results with improved physics. *Quarterly Journal of the Royal Meteorological Society*, 126: 1171–1190, 2000.

T. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Mathematics of Computation*, 34:473–497, 1980.

S. Massart, B. Pajot, A. Piacentini, and O. Pannekoucke. On the merits of using a 3D-FGAT assimilation scheme with an outer loop for atmospheric

situations governed by transport. *Monthly Weather Review*, 138:4509–4522, 2010.

G. Meurant and Z. Strakos. The lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numerica*, page 471–542, 2006.

I. Mirouze and A.Weaver. Representation of correlation functions in variational assimilation using an implicit diffusion operator. *Quarterly Journal of the Royal Meteorological Society*, 136:1421–1443, 2010.

I. Mirouze and A. Weaver. Representation of correlation functions in variational assimilation using an implicit diffusion operator. *Quarterly Journal of the Royal Meteorological Society*, 136:1421–1443, July 2010.

D. Montgomery and G. Runger. *Applied Statistics and Probability for Engineers*. John Wiley and Sons, 2007.

B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transaction on Automatic Control*, 26:17–32, 1981.

J.-D. Müller and P. Cusdin. On the performance of discrete adjoint CFD codes using automatic differentiation. *International Journal for Numerical Methods in Fluids*, 47:935–945, 2005.

S. Nash and A. Sofer. Block truncated-newton methods for parallel optimization. *Mathematical Programming*, 45:529–546, 1989.

L. Nazareth. A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms. *SIAM Journal on Numerical Analysis*, 16:794–800, 1979.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operation Research, 2006.

J. Nocedal, A. Sartenaer, and C. Zhu. On the behavior of the gradient norm in the steepest descent method. *Computational Optimization and Applications*, 22:5–35, 2002.

E. Ott, B. Hunt, I. Szunyogh, A. Zimin, E. Kostelich, M. Corazza, E. Kalnay, D. Patil, and J. Yorke. A local Ensemble Kalman filter for atmospheric data assimilation. *Tellus A*, 56:415–428, 2004.

M. Overton. *Numerical Computing with IEEE Floating Point Arithmetic*. SIAM, 2001.

D. Parrish and J. Derber. The national meteorological center's spectral statistical-interpolation analysis system. *Monthly Weather Review*, 120: 1747–1763, 1992.

D. T. Pham, J. Verron, and M.C. Roubaud. A singular evolutive extended Kalman filter for data assimilation in oceanography. *Journal of Marine Systems*, 16:323–340, 1998.

Samuel Pimentel. *Estimation of the Diurnal Variability of Sea Surface Temperatures using Numerical Modelling and the Assimilation of Satellite Observations*. PhD thesis, University of Reading, 2006.

A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Springer, 2000.

F. Rabier. Overview of global data assimilation developments in numerical weather-prediction centres. *Quarterly Journal of the Royal Meteorological Society*, 131:3215–3233, 2005.

F. Rabier and P. Courtier. Four-dimensional assimilation in the presence of baroclinic instability. *Quarterly Journal of the Royal Meteorological Society*, 118:649–672, July 1992.

F. Rabier, H. Jarvinen, E. Klinker, J.-F. Mahfouf, and A. Simmons. The ECMWF operational implementation of four-dimensional variational assimilation. I: Experimental results with simplified physics. *Quarterly Journal of the Royal Meteorological Society*, 126:1143–1170, 2000a.

F. Rabier, H. Järvinen, E. Klinker, J.-F. Mahfouf, and A. Simmons. The ECMWF operational implementation of four-dimensional variational assimilation. I: Experimental results with simplified physics. *Quarterly Journal of the Royal Meteorological Society*, 126:1143–1170, 2000b.

L. Richardson. *Weather Prediction by Numerical Process*. Cambridge University Press, 1922.

B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman filter : Particle filters for tracking applications*. Artech House Radar Library, 2004.

C. Robert, S. Durbiano, E. Blayo, J. Verron, J. Blum, and F.-X. Le Dimet. A reduced order strategy for 4D-var data assimilation. *Journal of Marine Systems*, 57:70–82, 2005.

C. Robert, E. Blayo, and J. Verron. Reduced-order 4D-Var: a preconditioner for the incremental 4D-Var data assimilation method. *Geophysical Research Letters*, 33:1–4, 2006.

D. Rozier, F. Birol, E. Cosme, P. Brasseur, J. M. Brankart, and J. Verron. A reduced-order Kalman filter for data assimilation in physical oceanography. *SIAM Review*, 49:449–465, 2007.

Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, second edition, 2008.

Y. Sasaki. An objective analysis based on the variational method. *Journal of the Meteorological Society of Japan*, 36:77–88, 1958.

D. Simon. *Optimal State Estimation*. Wiley-Interscience, 2006.

C. Snyder, T. Bengtsson, P. Bickel, and J. Anderson. Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, 12:4629–4640, 2008.

G. Stewart. *Matrix Algorithms Volume II : Eigensystems*. SIAM, 2001.

G. Strang and K. Borre. *Linear Algebra, Geodesy, and GPS*. Wellesley-Cambridge Press, 1997.

W. Sun and Y-X Yuan. *Optimization Theory and Methods*. Springer, 2006.

O. Talagrand and P. Courtier. Variational assimilation of meteorological observations with the adjoint vorticity equation. I: Theory. *Quarterly Journal of the Royal Meteorological Society*, 113:1311–1328, 1987.

M.-H. Tber, L. Hascoët, A. Vidard, and B. Dauvergne. Building the tangent and adjoint codes of the ocean general circulation model OPA with the automatic differentiation tool TAPENADE. Research report, INRIA, 2007.

C. Thornton. *Triangular Covariance Factorizations for kalman Filtering*. PhD thesis, University of California, 1976.

R. Todling and S. E. Cohn. Suboptimal schemes for atmospheric data assimilation based on the Kalman filter. *Monthly Weather Review*, 122:2530–2557, 1994.

V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7:1–25, 1993.

Y. Trémolet. Diagnostics of linear and incremental approximations in 4D-Var. *Quarterly Journal of the Royal Meteorological Society*, 130:2233–2251, 2004.

Y. Trémolet. Accounting for an imperfect model in 4D-Var. *Quarterly Journal of the Royal Meteorological Society*, 132:2483–2504, 2006.

J. Tshimanga. *On a Class of Limited Memory Preconditioners for Large-Scale Nonlinear Least Squares Problems*. PhD thesis, University of Namur, 2007.

J. Tshimanga, S. Gratton, A. Weaver, and A. Sartenaer. Limited-memory preconditioners, with application to incremental four-dimensional variational data assimilation. *Quarterly Journal of the royal Meteorological Society*, 134: 751–769, 2008.

Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly, and R. Marti. Scatter search and local NLP solvers: A multistart framework for global optimization. *Journal on Computing*, 19:328–340, 2007.

H. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge Monographs on Applied and Computational Mathematics, 2003.

M. Vancoppenolle, S. Bouillon, T. Fichefet, H. Goosse, O. Lecomte, M.A. Morales Maqueda, and G. Madec. LIM the Louvain-la-Neuve sea ice model. *Note du Pole de Modélisation, Institut Pierre-Simon Laplace*, 2012.

F. Veerse and J.N. Thépaut. Multiple-truncation incremental approach for four-dimensional variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 124:1889–1908, 1998.

B. Weare and J. Nasstrom. Examples of extended empirical orthogonal function analyses. *Monthly Weather Review*, 110:481–485, 1982.

A. Weaver and P. Courtier. Correlation modeling on the sphere using a generalized diffusion equation. *Quarterly Journal of the Royal Meteorological Society*, 127:1815–1842, 2001.

A. Weaver and S. Ricci. Constructing a background error correlation model using generalized diffusion operator. *Proceedings of the ECMWF Seminar Series on Recent Developments in Atmospheric and Ocean Data Assimilation"*, pages 327–340, 2003.

A. Weaver, J. Viliard, and D. Anderson. Three- and four-dimensional variational assimilation with an ocean general circulation model of the tropical pacific ocean. I: Formulation, internal diagnostics and consistency checks. *Monthly Weather Review*, 131:1360–1378, 2003.

A. Weaver, C. Deltel, E. Machu, S. Ricci, and N. Daget. A multivariate balance operator for variational ocean data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 131:3605–3625, 2005.

J. Whitaker and T. Hamill. Ensemble data assimilation without perturbed observations. *Monthly Weather Review*, 130:1913–1924, 2002.

A. Wiin-Nielsen. The birth of numerical weather prediction. *Tellus*, 43:36–52, 1991.

F. Zhang and E. Hancock. Graph spectral image smoothing using the heat kernel. *Pattern Recognition*, 41:3328 – 3342, 2008.

Appendices

Appendix A

Calculus

A.1 Matrix pseudo-inverse

A square matrix has an inverse if and only if its determinant is not equal to zero. The goal of matrix pseudo-inverse is to obtain a matrix that can serve as the inverse in some sense for a wider class of matrices than invertible ones. There are various kinds of matrix pseudo-inverses. The most widely known is called the Moore–Penrose pseudo-inverse (Golub and Van Loan, 1996).

Definition *For $\mathbf{A} \in \mathbb{R}^{n \times m}$, the Moore–Penrose pseudo-inverse, hereafter just called pseudo-inverse, is defined as the matrix $\mathbf{A}^+ \in \mathbb{R}^{m \times n}$ satisfying all of the following four criteria:*

- $\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A}$,
- $\mathbf{A}^+\mathbf{A}\mathbf{A}^+ = \mathbf{A}^+$,
- $(\mathbf{A}\mathbf{A}^+)^T = \mathbf{A}\mathbf{A}^+$,
- $(\mathbf{A}^+\mathbf{A})^T = \mathbf{A}^+\mathbf{A}$.

This pseudo-inverse exists and is unique for any matrix (Golub and Van Loan, 1996). Its construction is based on the rank decomposition of $\mathbf{A}$. Assume that $k \leqslant \min(n, m)$ denotes the rank of $\mathbf{A}$, then $\mathbf{A}$ can be decomposed as $\mathbf{A} = \mathbf{A}_1\mathbf{A}_2$, where $\mathbf{A}_1 \in \mathbb{R}^{n \times k}$ and $A_2 \in \mathbb{R}^{k \times m}$ are of rank k. The pseudo-inverse is given by

$$\mathbf{A}^+ = \mathbf{A}_2^+\mathbf{A}_1^+ = \mathbf{A}_2^T(\mathbf{A}_2\mathbf{A}_2^T)^{-1}(\mathbf{A}_1^T\mathbf{A}_1)^{-1}\mathbf{A}_1^T.$$

This general definition can be simplified for special matrix cases. We illustrate this for three special cases. If the matrix $\mathbf{A}$ is full column rank, i.e., the m columns of $\mathbf{A}$ are linearly independent, the pseudo-inverse is given by

$$\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T.$$

If the matrix $\mathbf{A}$ is full row rank, i.e., the n rows of $\mathbf{A}$ are linearly independent, the pseudo-inverse is given by

$$\mathbf{A}^+ = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}.$$

Finally, if $\mathbf{A}$ has orthonormal rows or columns, the pseudo-inverse is simply given by the transpose of the matrix,

$$\mathbf{A}^+ = \mathbf{A}^T.$$

To compute numerically the pseudo-inverse in a simple and accurate way, the Singular Value Decomposition (SVD) can be used. This decomposition is given by

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

where the columns of the orthogonal matrix $\mathbf{U} \in \mathbb{R}^{n\times n}$ are the left singular vectors, the columns of the orthogonal matrix $\mathbf{V} \in \mathbb{R}^{m\times m}$ are the right singular vectors and $\mathbf{\Sigma} \in \mathbb{R}^{n\times m}$ is a diagonal matrix containing the singular values of $\mathbf{A}$. The pseudo-inverse is then computed as

$$\mathbf{A}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^T,$$

where the matrix $\mathbf{\Sigma}^+$ is obtained by taking the inverse of each nonzero element on the diagonal, leaving the zeros in place, and transposing the resulting matrix. In numerical computations, only the diagonal elements larger than some small tolerance are taken to be nonzero.

A.2 Sherman–Morrison–Woodbury formula

The Sherman–Morrison–Woodbury (SMW) formula says that the inverse of a rank-k correction of some matrix can be computed by doing a rank-k correction to the inverse of the original matrix. Assuming a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ of rank n and a perturbation $\mathbf{UCV}$ of rank k with $\mathbf{C} \in \mathbb{R}^{k\times k}$ of rank k, $\mathbf{U} \in \mathbb{R}^{n\times k}$ and $\mathbf{V} \in \mathbb{R}^{k\times n}$, the SMW formula is given by

$$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{C}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U}\right)^{-1}\mathbf{V}\mathbf{A}^{-1}. \tag{A.1}$$

Note that in the above equation the matrix $\mathbf{C}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U}$ is assumed to be invertible. There is another useful equality which transforms the inverse of a matrix of rank n into the inverse of a matrix of rank k. It is given by

$$(\mathbf{A}^{-1} + \mathbf{U}\mathbf{C}^{-1}\mathbf{V})^{-1}\mathbf{U}\mathbf{C}^{-1} = \mathbf{A}\mathbf{U}(\mathbf{C} + \mathbf{V}\mathbf{A}\mathbf{U})^{-1}. \tag{A.2}$$

Assuming that $\mathbf{A}^{-1} + \mathbf{U}\mathbf{C}^{-1}\mathbf{V}$ and $\mathbf{C} + \mathbf{V}\mathbf{A}\mathbf{U}$ are invertible, the proof is straightforward using simple linear algebra calculus.

Proof. — From the equality

$$\mathbf{U} + \mathbf{U}\mathbf{C}^{-1}\mathbf{V}\mathbf{A}\mathbf{U} = \mathbf{U} + \mathbf{U}\mathbf{C}^{-1}\mathbf{V}\mathbf{A}\mathbf{U}$$

and performing two different factorizations, we obtain

$$\mathbf{U}\mathbf{C}^{-1}(\mathbf{C} + \mathbf{V}\mathbf{A}\mathbf{U}) = (\mathbf{A}^{-1} + \mathbf{U}\mathbf{C}^{-1}\mathbf{V})\mathbf{A}\mathbf{U}.$$

Left-multiplying by $(\mathbf{A}^{-1}+\mathbf{U}\mathbf{C}^{-1}\mathbf{V})^{-1}$ and right-multiplying by $(\mathbf{C}+\mathbf{V}\mathbf{A}\mathbf{U})^{-1}$ each side of the equation leads to

$$\begin{aligned}(\mathbf{A}^{-1} + \mathbf{U}\mathbf{C}^{-1}\mathbf{V})^{-1}&\mathbf{U}\mathbf{C}^{-1}(\mathbf{C} + \mathbf{V}\mathbf{A}\mathbf{U})(\mathbf{C} + \mathbf{V}\mathbf{A}\mathbf{U})^{-1} \\ &= (\mathbf{A}^{-1} + \mathbf{U}\mathbf{C}^{-1}\mathbf{V})^{-1}(\mathbf{A}^{-1} + \mathbf{U}\mathbf{C}^{-1}\mathbf{V})\mathbf{A}\mathbf{U}(\mathbf{C} + \mathbf{V}\mathbf{A}\mathbf{U})^{-1},\end{aligned}$$

which gives the desirable result by simplification. ∎

A.3 Derivative rules

We recall some useful results of matrix calculus and derivative rules. Firstly, suppose that $\mathbf{x} \in \mathbb{R}^n$ is a column vector and that $f : \mathbb{R}^n \to \mathbb{R} : \mathbf{x} \rightsquigarrow f(\mathbf{x})$ is a scalar function of $\mathbf{x}$. The derivative of a scalar function with respect to a vector is given by the gradient of the function,

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_1} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_n} \end{pmatrix} \tag{A.3}$$

which is assumed to be a column vector. Using (A.3), we can compute the derivative of a dot product between $\mathbf{x}$ and a constant vector $\mathbf{y} \in \mathbb{R}^n$,

$$\nabla_{\mathbf{x}} \left(\mathbf{x}^T\mathbf{y}\right) = \mathbf{y}, \tag{A.4}$$

and the derivative of a quadratic function $\mathbf{x}^T\mathbf{B}\mathbf{x}$ with $\mathbf{B} \in \mathbb{R}^{n\times n}$,

$$\nabla_{\mathbf{x}} \left(\mathbf{x}^T\mathbf{B}\mathbf{x}\right) = \mathbf{B}\mathbf{x} + \mathbf{B}^T\mathbf{x}. \tag{A.5}$$

If the matrix $\mathbf{B}$ is symmetric, the last result is simply $2\mathbf{B}\mathbf{x}$. The derivative of a quadratic function shifted by a constant vector $\mathbf{b} \in \mathbb{R}^n$ is given by

$$\nabla_{\mathbf{x}} \left((\mathbf{x} - \mathbf{b})^T\mathbf{B}(\mathbf{x} - \mathbf{b})\right) = \mathbf{B}(\mathbf{x} - \mathbf{b}) + \mathbf{B}^T(\mathbf{x} - \mathbf{b}), \tag{A.6}$$

which is equal to $2\mathbf{B}(\mathbf{x} - \mathbf{b})$ if the matrix $\mathbf{B}$ is symmetric. A last useful rule can be deduced,

$$\nabla_{\mathbf{x}} \left((\mathbf{G}\mathbf{x} - \mathbf{b})^T\mathbf{B}(\mathbf{G}\mathbf{x} - \mathbf{b})\right) = \mathbf{G}^T\mathbf{B}(\mathbf{G}\mathbf{x} - \mathbf{b}) + \mathbf{G}^T\mathbf{B}^T(\mathbf{G}\mathbf{x} - \mathbf{b}), \tag{A.7}$$

where $\mathbf{G} \in \mathbb{R}^{n \times n}$ is a constant matrix. If the matrix $\mathbf{B}$ is symmetric, this result is simply $2\mathbf{G}^T\mathbf{B}(\mathbf{G}\mathbf{x} - \mathbf{b})$.

Secondly, one can be interested by computing the derivative of a vector with respect to another vector. To this aim, suppose that

$$F : \mathbb{R}^n \to \mathbb{R}^m : \mathbf{x} \rightsquigarrow : F(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix},$$

then the derivative of F is given by the Jacobian matrix which is defined as

$$\nabla_{\mathbf{x}} F(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial \mathbf{x}_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial \mathbf{x}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial \mathbf{x}_1} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial \mathbf{x}_n} \end{pmatrix}.$$

Thirdly, it is possible to calculate the derivative of a function depending on a matrix. Suppose that $h : \mathbb{R}^{n \times p} \to \mathbb{R} : \mathbf{X} \rightsquigarrow h(\mathbf{X})$ is a scalar function of the matrix $\mathbf{X}$. The derivate of h with respect to $\mathbf{X}$ is given by

$$\nabla_{\mathbf{X}} h(\mathbf{X}) = \begin{pmatrix} \frac{\partial h(\mathbf{X})}{\partial \mathbf{X}_{11}} & \cdots & \frac{\partial h(\mathbf{X})}{\partial \mathbf{X}_{1p}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h(\mathbf{X})}{\partial \mathbf{X}_{n1}} & \cdots & \frac{\partial h(\mathbf{X})}{\partial \mathbf{X}_{np}} \end{pmatrix}.$$

Assuming two constant vector $\mathbf{a} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^p$, we obtain from the definition that

$$\nabla_{\mathbf{X}} \left(\mathbf{a}^T \mathbf{X} \mathbf{b} \right) = \mathbf{a}\mathbf{b}^T. \tag{A.8}$$

It is also possible to deduce the derivative of the trace of a product of matrices. Suppose that $\mathbf{C} \in \mathbb{R}^{p \times p}$ is a constant matrix, then the trace of $\mathbf{X}\mathbf{C}\mathbf{X}^T$ is given by

$$h(\mathbf{X}) = tr(\mathbf{X}\mathbf{C}\mathbf{X}^T) = \sum_{i=1}^{n} \sum_{j=1}^{p} \sum_{k=1}^{p} \mathbf{X}_{ik} \mathbf{C}_{kj} \mathbf{X}_{ij}$$

and we have

$$\nabla_{\mathbf{X}} \left(tr(\mathbf{X}\mathbf{C}\mathbf{X}^T) \right) = \mathbf{X}\mathbf{C}^T + \mathbf{X}\mathbf{C}. \tag{A.9}$$

If the matrix $\mathbf{C}$ is symmetric, this results is simply $2\mathbf{X}\mathbf{C}$.

Finally, it is possible to derive a matrix with respect to another matrix. Since the definition of this differentiation process involves heavier notations, only one useful example is given. Assume that $\mathbf{X} \in \mathbb{R}^{n \times p}$ and that $\mathbf{D} \in \mathbb{R}^{p \times n}$, we have

$$\nabla_{\mathbf{X}} \left(\mathbf{X}\mathbf{D} \right) = \mathbf{D}^T. \tag{A.10}$$

Other derivative formula with further calculation details can be found in Simon (2006).

Appendix B

Theory of constrained optimization

We recall an interesting and useful result about the minimization of a function under equality constraints. Assume that we have the following minimization problem

$$\left\{ \begin{array}{l} \min\limits_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \\ s.t. \quad \mathbf{c}_i(\mathbf{x}) = 0, \ i = 1, \ldots, m, \end{array} \right. \tag{B.1}$$

where f and $\mathbf{c}_i$, $i = 1, \ldots, m$ are scalar functions. We are interested by finding a strict local solution $\mathbf{x}^*$ of (B.1), i.e., a feasible point $\mathbf{x}^*$ with a neighborhood $\mathcal{N}$ such that $f(\mathbf{x}) > f(\mathbf{x}^*)$ for all feasible points $\mathbf{x} \neq \mathbf{x}^*$ in the neighborhood. The theorem of second-order sufficient optimality conditions gives conditions to obtain such a solution. Before presenting it, we need to define the Lagrangian function

$$\mathcal{L} : \mathbb{R}^{n+m} \to \mathbb{R} : (\mathbf{x}, \boldsymbol{\lambda}) \rightsquigarrow \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i=1}^{m} \boldsymbol{\lambda}_i \mathbf{c}_i(\mathbf{x}), \tag{B.2}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the Lagrangian multiplier vector.

Theorem B.1 *Suppose that for some feasible point $\mathbf{x}^* \in \mathbb{R}^n$ there is a Lagrange multiplier vector $\boldsymbol{\lambda}^*$ such that the Karush–Kuhn–Tucker (KKT) conditions*

$$\begin{array}{rcll} \nabla_{\mathbf{x}} L(\mathbf{x}^*, \boldsymbol{\lambda}^*) & = & 0 & \\ \mathbf{c}_i(\mathbf{x}) & = & 0 & i = 1, \ldots, m \end{array}$$

are satisfied. Suppose also that

$$\mathbf{w}^T \nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \mathbf{w} > 0$$

for all $\mathbf{w} \in \mathbb{R}^n$ *such that* $\nabla_x \mathbf{c}_i(\mathbf{x}^*)^T \mathbf{w} = 0$. *Then* $\mathbf{x}^*$ *is a strict local solution of (B.1).*

There are other theorems which treat optimality conditions of optimization problems. They are not presented in this work since they are not used but can be found in Nocedal and Wright (2006).

Theorem B.1 which states second-order sufficient optimality conditions for the problem (B.1) can be easily generalized for problems where the objective function and the constraints depend on a matrix instead of a vector. Assume that

$$\begin{aligned} h &: \mathbb{R}^{n \times p} \to \mathbb{R} : \mathbf{X} \rightsquigarrow h(\mathbf{X}) \\ \mathbf{d}_i &: \mathbb{R}^{n \times p} \to \mathbb{R} : \mathbf{X} \rightsquigarrow \mathbf{d}_i(\mathbf{X}), \qquad i = 1, \ldots, m, \end{aligned}$$

The solution of

$$\begin{cases} \min\limits_{\mathbf{X} \in \mathbb{R}^{n \times p}} h(\mathbf{X}) \\ s.t. \quad \mathbf{d}_i(\mathbf{X}) = 0, \ i = 1, \ldots, m, \end{cases} \tag{B.3}$$

is given by Theorem $B.1$ where the derivatives with respect to a vector are replaced by derivatives with respect to a matrix (see Appendix A.3). In the following, we illustrate this generalization in the particular case of

$$\begin{cases} \min\limits_{\mathbf{X} \in \mathbb{R}^{n \times p}} h(\mathbf{X}) \\ s.t. \quad \mathbf{X}\mathbf{A} - \mathbf{B} = 0, \end{cases} \tag{B.4}$$

where $\mathbf{A} \in \mathbb{R}^{p \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times n}$ are constant matrices. One can remark that the constraints are given under the form of a matrix equation which defines implicitly n^2 linear constraints on the values of $\mathbf{X}$. To apply Theorem B.1, the linear constraints are stated independently such that

$$d_{kl}(\mathbf{X}) = \sum_{m=1}^{p} \mathbf{X}(k,m)\mathbf{A}(m,l) - \mathbf{B}(k,l), \quad k,l = 1, \ldots, n,$$

where the notation (i,j) designates the matrix element at the ith row and the jth column. The Lagrangian function (B.2) is then given by

$$\mathcal{L}(\mathbf{X}, \boldsymbol{\Lambda}) = h(\mathbf{X}) - \sum_{k=1}^{n} \sum_{l=1}^{n} \boldsymbol{\Lambda}(k,l) \left[\sum_{m=1}^{p} \mathbf{X}(k,m)\mathbf{A}(m,l) - \mathbf{B}(k,l) \right], \tag{B.5}$$

where $\mathbf{\Lambda} \in \mathbb{R}^{n \times n}$ is the Lagrangian multiplier matrix. It can be rewritten in a shorter way as

$$\mathcal{L}(\mathbf{X}, \mathbf{\Lambda}) = h(\mathbf{X}) - \sum_{l=1}^{n} \mathbf{\Lambda}(:, l)^T \left[\mathbf{X}\mathbf{A}(:, l) - \mathbf{B}(:, l)\right], \tag{B.6}$$

where $(:, l)$ denotes the lth column of a matrix. Using the formula (A.8), the differentiation of (B.6) with respect to $\mathbf{X}$ is then given by

$$\begin{aligned} \nabla_{\mathbf{X}} \mathcal{L}(\mathbf{X}, \mathbf{\Lambda}) &= \nabla_{\mathbf{X}} h(\mathbf{X}) - \sum_{j=1}^{n} \mathbf{\Lambda}(:, j)\mathbf{A}(:, j)^T \\ &= \nabla_{\mathbf{X}} h(\mathbf{X}) - \mathbf{\Lambda}\mathbf{A}^T \end{aligned}$$

and the final KKT conditions are

$$\begin{aligned} \nabla_{\mathbf{X}} h(\mathbf{X}) - \mathbf{\Lambda}\mathbf{A}^T &= 0 \\ \mathbf{X}\mathbf{A} - \mathbf{B} &= 0. \end{aligned} \tag{B.7}$$